Lokalizacija in osnovna navigacija mobilnega robota s pomočjo vizualnega učenja in razpoznavanja

Matej Artač

Magistrska naloga
predložena
Fakulteti za računalništvo in informatiko
Univerze v Ljubljani
kot delna izpolnitev pogoja za pridobitev naslova
magister računalništva in informatike

Ljubljana, 2003

Mentor:
prof. dr. Aleš Leonardis

Magistrska naloga je bila izdelana pod mentorstvom prof. dr. Aleša Leonardisa in je last Fakultete za računalništvo in informatiko v Ljubljani. Za objavljanje in uporabo rezultatov magistrskega dela je potrebno soglasje zgoraj omenjene ustanove.

Besedilo je oblikovano s sistemom za pripravo besedil LaTeX.

Povzetek

Računalniški vid prevzema vedno večjo vlogo na področju mobilne robotike. Vizualni senzorji uspešno dopolnjujejo ali v celoti nadomeščajo uporabo klasičnih senzorjev. Zlasti dobre lastnosti imajo panoramski senzorji, ki z enim pogledom zajamejo izgled prizora v vseh smereh. Ker se ta izgled spreminja s spreminjanjem lokacije, s katere gledamo na prizor, nam vsako lokacijo opisuje svoj pogled. Predpostavljamo lahko, da je možno iz pogleda ugotoviti lokacijo, s katere dobimo omenjeni pogled. Ta predpostavka je osnova vizualne lokalizacije ter navigacije.

Predpogoj za izvedbo vizualne lokalizacije in navigacije je poznavanje množice pogledov iz lokacij v prostoru. Te med raziskovanjem zbiramo v obliki slik. Z vizualnim učenjem iz teh slik dobimo model prostora in predstavitve slik. Uveljavljen način vizualnega učenja vključuje uporabo metode glavnih komponent, ki slike predstavi na kompakten način in omogoči enostavno primerjanje podobnosti med njimi. Ker pa želimo, da učenje ostane odprto in da je sposobno model dopolniti z novimi slikami, smo uporabili postopno metodo glavnih komponent. Zanjo smo razvili postopek sprotne obravnave predstavitev slik, ki poskrbi, da izvirnih slik ni potrebno hraniti v pomnilniku. Dobljeni postopek je sposoben obdelati bistveno večje množice slik od osnovnega postopka, hkrati pa je izvedljiv na mobilnih platformah z omejenim pomnilnikom.

Vizualno razpoznavo smo uporabili v vlogi vizualne lokalizacije. Opisali smo postopek, ki uporablja lokalizacijo pri ugotavljanju trenutne lokacije ter vodenju navigacije skozi raziskani prostor do izbranega cilja. Predstavljene postopke smo implementirali ter njihovo uporabnost prikazali z empiričnimi poskusi na laboratorijskem mobilnem robotu.

Abstract

Computer vision is taking an increasingly important role in the field of mobile robotics. A number of researches show that the usage of visual sensors can successfully complement or even substitute the classic sensors. The results are particularly good if we use an omnidirectional sensor which captures the whole panorama of the scene by a single view. Since the view changes when we vary the location from where we capture it, we can use the appearance of each view to describe the corresponding location. It should therefore be possible to identify the location from its appearance, i.e., to perform a visual localisation.

In order for the localisation to work we first have to collect the visual information of the environment in the form of a set of images. Visual learning transforms these images into a model of the environment which describes the distinct visual information of each particular location. The method that has often been proven suitable for this process employs Principal Component Analysis (PCA). PCA builds a compact representation of the input data which allows for a simple way of comparing the data. However, since we require our system to perform on-line and be able to update the model with new images, we prefer the incremental method of computing PCA instead. In this work we propose a method which considerably reduces the amount of needed memory for the method, thus rendering it feasible for mobile platforms with limited memory.

We describe the application of the visual learning and recognition for visual localisation and basic navigation. These methods can build a map of the explored environment, and then use it along with visual cues only to localise itself and navigate to a destination within the environment. We show the performance of the methods with experiments on an in-door mobile robot.

Kazalo

Slike

Tabele

Poglavje 1

Uvod

1.1 Vid, učenje, navigacija

Človek ima nabor čutov, ki se med seboj dopolnjujejo. S pomočjo dražljajev, ki jih ti čuti zaznavajo, dobimo predstavo o dogajanju okrog nas ter o posledicah naših dejanj. Izmed njih je vid najbolj kompleksen in hkrati najpomembnejši čut. Količina informacij, ki jo dobimo preko njega, je izjemna.

Vid nam omogoča zaznavanje okolice okoli nas. S pomočjo kompleksnih zavednih ali nezavednih miselnih procesov iz vidnega dražljaja sklepamo na lastnosti in konfiguracijo prostora, kjer se nahajamo, ter ugotovimo svoj položaj v tem okolju. Če smo v tem prostoru prvič, si ga zapomnimo, ugotovitve pa uporabimo, ko prostor zapuščamo in ko se kasneje tja spet vrnemo.

Če za gibanje uporabimo hojo, nam za samo koordinacijo gibov okončin pomagajo čutila, kot sta čut za dotik ter ravnotežje. Vendar pa smo sposobni gibanja tudi na številne načine, kjer ni neposredne zveze med premiki in delom, ki ga opravljajo mišice. Takšno gibanje je lahko npr. plavanje, kjer na premik telesa vpliva tudi vodni tok, ali kolesarjenje, kjer z različnimi prestavnimi razmerji ob isti hitrosti vrtenja pedal dosežemo različne hitrosti vožnje. Svoj segment gibanja pa vključuje širok spekter vozil, ki uporabljajo lastne načine pogona, ljudje pa vozila samo krmilimo. Skrajni primer predstavlja navidezna resničnost, kjer zremo skozi oči navideznega lika, ki se giblje po navidezni pokrajini, krmilimo ga pa s pritiski na tipke.

Vsem naštetim načinom gibanja je skupno to, da s pomočjo vizualne informacije spoznavamo prostor, v katerem se nahajamo, ga skušamo na globalnem nivoju umestiti v že znan okvir predhodno obiskanih prostorov, ter s pomočjo razpoložljivih informacij načrtovati nadaljnje akcije. Na bolj lokalnem nivoju pa s pomočjo vizu-

alne informacije sklepamo tudi na posledice preteklih dejanj. O tem, kako ključnega pomena je vid za opravljanje omenjenih nalog, se lahko prepričamo z začasno deprivacijo vizualnih dražljajev: v popolni temi postane naloga hoje iz točke A v točko B občutno težji problem, četudi skušamo priti iz svoje spalnice v dnevno sobo.

Vendar pa sama sposobnost zaznavanja vizualnih informacij ni dovolj za uspešno navigacijo. Potapljač bo v jezeru, ki ga pokriva debela plast ledu, ob povratku brez vrvi zelo težko našel odprtino, kjer je potop začel. Podobno se je precej enostavno izgubiti v puščavi, globokem gozdu ali blodnjaku ulic naselja. Podvodni svet med gladino in dnom ter puščava sta primera okolja, ki ne nudita dovolj stanovitnih vizualnih opornih točk, te pa so predpogoj, da lahko samo s pogledom na okolje sklepamo na to, kje smo. Gozd in naselje pa za tujca predstavljata preobilico informacij, ki se mu ne zdijo dovolj značilne za uspešno navigacijo.

Razlika med tujcem in domačinom nekega kraja je ta, da ima slednji urejeno predstavo o okolju, ki vključuje prostorske odnose med posameznimi predeli okolja. Tujec se lahko v novem okolju znajde tako, da svojo predstavitev okolja prinese v obliki zemljevida. Zemljevid je koherentna, vendar tudi nekoliko abstraktna predstavitev okolja, ki od uporabnika zahteva interpretacijo ter zavestno sklepanje o lokaciji na zemljevidu glede na zgodovino vizualnih dražljajev. Če se sklepanje izkaže za pravilno ter posledično navigacija uspe, si uporabnik to zapomni, nauči se nove poti skozi okolje ter tako zgradi mentalni zemljevid.

Vid torej igra zelo veliko vlogo pri delovanju bioloških sistemov in v življenju ljudi. Vedno bolj priznano vlogo pa dobiva tudi, ko obravnavamo tehnične sisteme. Med gradnjo umetnih sistemov, za katere si želimo, da ravnajo samostojno in inteligentno, nam zgledi iz živalskega sveta ter človekovega obnašanja pogosto služijo kot navdih. Zato tudi ko obravnavamo tematiko raziskovanja nekega prostora in navigacije po njem, resno razmišljamo o možnosti uporabe vizualne informacije v tem procesu.

1.2 Mobilni roboti in računalniški vid

Na tržišču je moč dobiti širok spekter mobilnih robotov, ki se razlikujejo po svojih sposobnostih, opremi in načinu delovanja. Številni prototipi so lahko rezultat tudi lastne izdelave. Nekateri od njih so radijsko vodeni in zahtevajo ročno ali kakšno drugo dalinjsko upravljanje, avtonomni roboti pa so opremljeni še s procesno enoto ali računalnikom, ki lahko krmili celotni sistem. Za nas so zanimivi predvsem slednji, saj omogočajo programiranje vedenja, ki ni vezano na bližino bazne ali krmilne

postaje.

Mobilnost robotom omogočajo motorji, ki poganjajo kolesa ali posebne noge. Procesna enota krmili motorje tako, da jim podaja napetost na vhodu, ali pa dostopa do vmesne procesne enote, ki dobi na vhodu želeno smer in hitrost, kar potem pretvori v ustrezne ukaze motorju. Poleg pogona, ki omogoča akcijo, pa sistem na robotu potrebuje še povratne informacije o gibanju ter okolici, torej neke vrste umetna čutila. Ta imajo obliko aktivnih in pasivnih senzorjev.

Nabor senzorjev nam omogoča obilico numeričnih podatkov, ki nam ob smiselnem kombiniranju omogočijo spremljanje dogajanja okrog robota, ocenjevanje predhodnih ter načrtovanje naslednjih akcij. Pri vsakem od prebranih podatkov pa je treba upoštevati, da senzorji niso absolutno natančni in da se med seboj razlikujejo po stopnji zanesljivosti vrnjenih vrednosti.

Aktivni senzorji ugotavljajo oddaljenost od predmetov v neposredni okolici. Ti senzorji imajo aktivno komponento, ki v prostor oddaja neko energijo, ki se od bližnjih predmetov odbije. Amplitudo in časovni zamik meri pasivni del senzorja, ki nato izračuna oceno oddaljenosti predmetov. Tipični predstavniki aktivnih senzorjev nižje ločljivosti v prostor oddajajo zvok ali infrardečo svetlobo, višjo ločljivost pa dosegamo z laserskimi žarki.

Pasivni senzorji zgolj zbirajo energijo iz okolice, ne da bi jo sami oddajali. Izmed njih posebna vrsta senzorjev spremlja notranje stanje robota. Sistem motorjev za gibanje robota običajno vključuje notranje spremljanje premikov in vrtljajev pogonskih koles ter s tem povratno informacijo o trenutni (kotni) hitrosti gibanja. Z akumulacijo teh podatkov dobimo podatke o položaju in orientaciji robota v metričnem prostoru, imenovane tudi odometrija. V robota lahko vgradimo še senzorje, ki s pomočjo vztrajnosti ali magnetnih silnic Zemlje določajo orientacijo robota. Fizični stik s predmeti in trke zaznavajo stikala na ohišju robota, ki obveščajo o trkih robota.

Med pasivnimi senzorji so najbolj zanimivi tisti, ki omogočajo vizualno zaznavo. Video kamere so lahko nameščene statično na ohišje robota, s čimer omogočajo zaznavanje pogleda, ki je odvisen samo od poze robota. Število kamer je pri tem odvisno od naloge, ki jo sistem opravlja, in vrste informacij, ki jih sistem potrebuje. Poleg tega so lahko nameščene na gibljivi in vodljivi roki, kar omogoča zajemanje pogledov, ki so manj odvisni od robota. S takim sistemom lahko dosežemo t.i. aktivni vid, kjer sistem kamere, mobilne roke in programske opreme tvorno izbira pogled glede na vsebino opazovanega prizora in naloge, ki naj bi jih robot opravljal. Kamere se med seboj razlikujejo tudi po tem, kakšen spekter svetlobe zajemajo (npr. vidni

spekter, infrardeči) ter kakšna je geometrija nastanka slike (npr. perspektivna, panoramska kamera).

Podatke iz vizualnih senzorjev obdelujejo postopki, s katerimi se ukvarja računalniški vid. Ti algoritmi iščejo informacije, ki omogočajo krmiljenje preostalih procesov. Mobilni roboti potrebujejo predvsem informacije o svoji legi, položaju in o prostoru, v katerem se nahaja.

1.2.1 Zemljevidi

Zemljevid je skupek prostorsko povezanih podatkov, ki nosi predstavitev o konfiguraciji prostora, po katerem se gibljemo. Običajno prikazuje, kateri deli prostora so prehodni in kateri neprehodni, to pa omogoča navigacijo med dvema točkama v prostoru. Če se nahajamo na neznani lokaciji, pa naj bi bilo s pomočjo podatkov o okolici trenutne lokacije moč umestiti trenutno lokacijo na zemljevid.

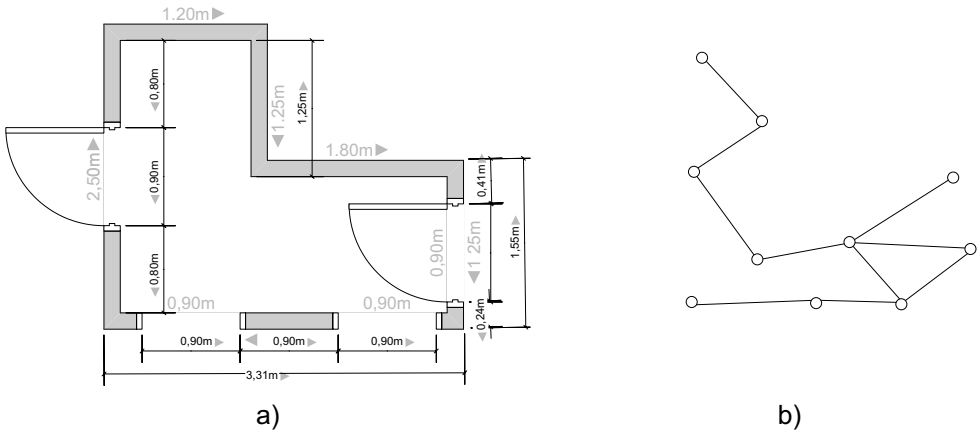
Zemljevid lahko pripravimo ročno in ga sistemu podamo kot vhod. Precej bolj zanimiva pa je aplikacija, v kateri robot sam razišče prej neznan prostor in si zgradi predstavo o njem. Ta naloga je razmeroma zahtevna, saj moramo med gradnjo upoštevati napake in šum pri vhodnih podatkih, zaznavati in odpravljati odstopanja.

Kako zemljevid dejansko izgleda, je odvisno od vrste podatkov, ki jih uporabimo pri zaznavanju okolja.

- Če nam podatki podajajo obliko ali oddaljenost od robota do predmetov v neposredni bližini (zvočni in infrardeči senzorji, stereo vid), zgradimo *metrični zemljevid*. Takšen zemljevid je še najbližji človekovi percepciji zemljevida, saj podaja neke vrste tloris preiskanega prostora. Primer metričnega zemljevida prikazuje slika 1.1 a).

- *Topološki zemljevid* pa predstavlja okolje kot graf, katerega vozlišča predstavljajo posamezne lokacije ali večje prostore, po katerih se lahko robot giblje, povezave med vozlišči pa nakazujejo pare prostorov, ki so med seboj neposredno dostopni (npr. dve sobi, ki ju ločujejo odprta vrata). Slika 1.1 b) predstavlja ilustracijo topološkega zemljevida.

Seveda pa pri izbiri vrste zemljevida nismo neposredno vezani z vrsto senzorjev, ampak lahko vsak metrični zemljevid spremenimo v topološkega. Primerni način za to pretvorbo je, da kot posamezno vozlišče identificiramo večjo površino, kjer je vsak par točk med sabo neposredno dosegljiv. Rezultat je topološki zemljevid, ki



Slika 1.1: Primer metričnega zemljevida (a) in topološkega zemljevida (b).

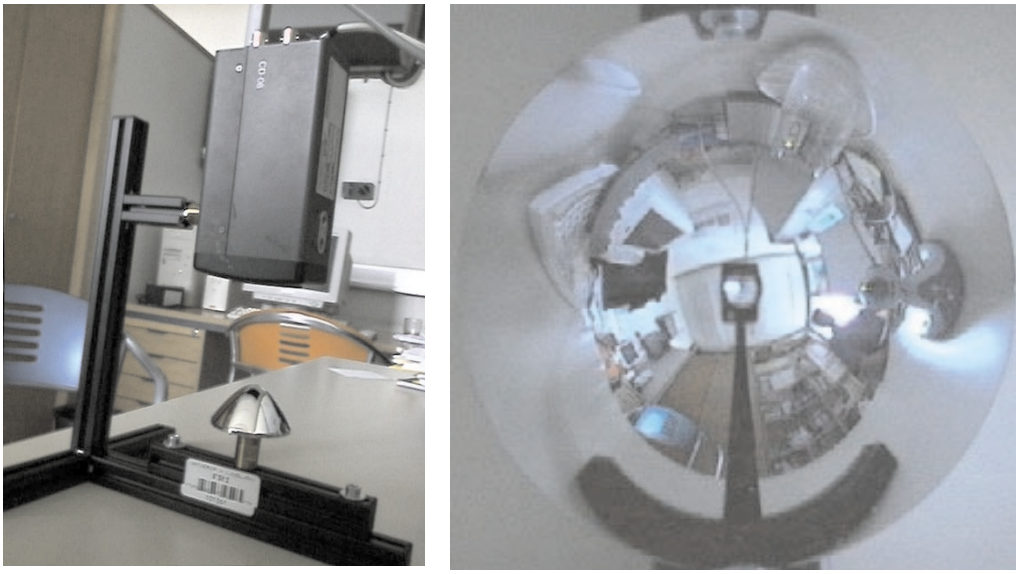
z vozlišči predstavlja večje dele prostora, s povezavami pa označuje prehode med njimi.

1.2.2 Panoramski senzor

Če na objektiv namestimo posebno lečo, lahko pridobimo panoramski pogled: zorni kot kamere se poveča do te mere, da dobimo vseh 360° prizora okrog osi projekcije običajne kamere, ki sestavlja sistem. Podoben učinek dosežemo, če (običajno) kamero usmerimo v zrcalo, katere površina opisuje ploskev druge stopnje (površina hiperbole ali parabole). Takšen sistem imenujemo tudi katadioptična kamera. Slika 1.2 a) prikazuje primer kamere usmerjene v hiperbolično zrcalo, slika 1.2 b) pa je pogled, kot ga zabeleži kamera s slike a).

Slika panoramskega senzorja na sicer pravokotni sliki zaseda okroglo področje, kot to prikazujeta sliki 1.2 b) in 1.3 a). Ker je relevantna slikovna informacija le v krožnem delu, običajno tisti del slike *razvijemo*. To naredimo s preslikavo polarnih koordinat v kartezične. Slika 1.3 ponazarja to preslikavo. Sliki a) in c) sta izvirni sliki, pri čemer je prva primer slike resničnega prizora, druga pa je umetno ustvarjena slika velikosti 50×50 slikovnih elementov. Sliki b) in d) prikazujeta rezultat preslikave. Barvni krožnici iz slik a) in c) se preslikata v vodoravni daljici na slikah b) in d), pri čemer zunanja krožnica postane zgornja vrstica nove slike, notranja krožnica pa spodnja vrstica. Podobno kot se krožnice na prvotni sliki preslikajo v vodoravne črte na cilindrični sliki, se radialne daljice prvotne slike, preslikajo v navpične črte.

Notranja krožnica ima majhen polmer, zato je njen obseg majhen. Ker pa se vse



a) b)

Slika 1.2: Panoramski senzor (a) in slika, ki jo zajamemo s tem senzorjem (b).

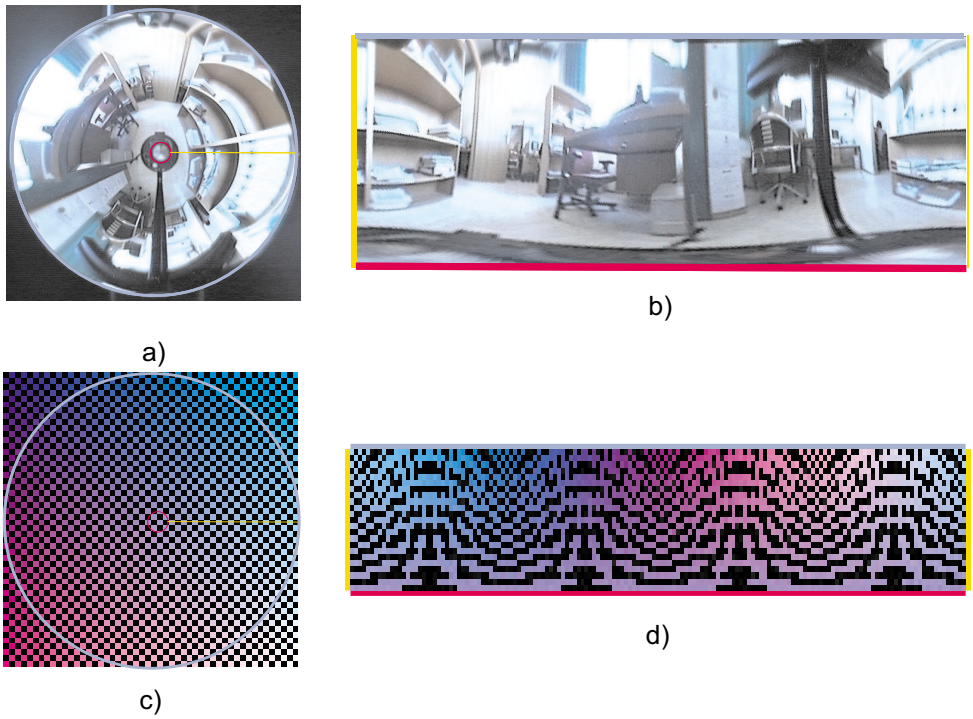
krožnice preslikajo v vrstice enake dolžine, lahko pri spodnjih vrsticah slike 1.3 d) opazimo značilen raztegnjeni vzorec. Slikovne elemente na tem delu lahko zgladimo ali interpoliramo, vendar s tem ne pridobimo novih informacij. Zato je smiselno senzor obrniti tako, da je v zgornjem delu slike največ ključnih delov prizora, saj tako dobimo več pomembnih podrobnosti.

1.3 Cilj naloge

V uvodnem razmisleku smo poudarili pomen vida pri spoznavanju okolja, ugotavljanju naše trenutne lokacije v okolju ter njeno vlogo pri navigaciji skozi okolje. Zato želimo, da bi podobni principi delovali tudi, če zgradimo umetni sistem, ki je sposoben samostojnega premikanja in vizualne zaznave okolice.

Avtonomni mobilni roboti imajo vgrajen računalnik, ki s proizvajalčevimi programskimi vmesniki omogoča preprost dostop uporabnikovega programa do pogona in senzorjev. Tako lahko odmislimo tehnične podrobnosti krmiljenja robota in ravnanja s senzorji, ker je ta del poenostavljen na nivo klicev nekaj funkcij. Osredotočimo se lahko na izvedbo vizualnega učenja, izgradnje predstavitve prostora ter uporabe tega znanja pri navigaciji.

Za izvedbo vizualnega učenja in razpoznavanja uporabimo dognanja s področja



Slika 1.3: Slika v hiperboličnem zrcalu (a) in njen razvoj v cilindrično sliko (b); umetna slika mreže (c) in razvoj označenega kolobarja (d).

računalniškega vida. Zanima nas predvsem tisti del vizualnega učenja, ki obravnava *poglede* kot osnovno enoto znanja. Takšen pristop nam omogoča uporabo slik v svoji osnovni obliki, s čimer se izognemo zapletenim postopkom analize slike, iskanja značilk na njih ter poskusom njihovega sledenja skozi čas. Poleg tega se na slikah ne osredotočimo samo na izbrano podmnožico podrobnosti, ampak obravnavamo sliko kot celostno vizualno informacijo. Pri izbiri postopka pa moramo biti pazljivi, saj smo na platformi mobilnega robota omejeni tako s pomnilnikom kot s hitrostjo procesiranja. Tudi če procesiranje prenesemo na drug računalnik, lahko postane ozko grlo brezvrvična komunikacija med namizno in mobilno enoto.

Ko imamo metodo vizualnega učenja izbrano, jo želimo uporabiti na mobilnem robotu. To bi naredili tako, da bi robot med raziskovanjem beležil poglede na različnih lokacijah, ter z učenjem vzpostavil zvezo med izgledom lokacije in koordinatami te lokacije. Omenjeno zvezo bi interno predstavil z zemljevidom raziskanega prostora. Zemljevid bi uporabil vsakič, ko bi želel preveriti svojo trenutno lokacijo. To bi naredil tako, da bi med učenjem vzpostavljeno zvezo med lokacijo in njenim izgledom izkoristil v obratni smeri: ugotovil bi, kateri del zemljevida je najbolj

podoben trenutnemu izgledu, kar bi mu dalo koordinate trenutne lokacije.

Zemljevid nam torej pomaga pri umeščanju naše trenutne lokacije v prostor, poleg tega pa lahko ugotavljamo tudi prostorske odnose različnih lokacij v prostoru. Če imamo znano svojo trenutno lokacijo, poznamo izhodišče, umeščeno na zemljevidu. Ko na zemljevidu izberemo še ciljno točko, lahko med tem parom poiščemo pot. Navigacija robota je nato proces uresničevanja te poti. Če hočemo, da navigacija uspe, moramo na poti sproti ugotavljati morebitne napake v načrtu poti ali v izvajanju vožnje ter jih odpravljati. Poleg tega lahko navigacijo uspešno zaključimo šele, ko ugotovimo dovolj veliko podobnost trenutne lokacije s ciljno lokacijo. Za delujočo navigacijo pa potrebujemo postopek, ki zna izvesti osnovni korak navigacije. Ta korak je sestavljen iz identifikacije trenutne lokacije, izbire smeri do podcilja, ki robota pripelje bližje cilju in je do njega mogoče priti, ne da bi robot naletel na oviro, nato vožnje do podcilja in preverjanje lokacije na koncu koraka.

Želimo torej zgraditi metodo lokalizacije in osnovne navigacije, ki ga vodi vizualno zgrajen zemljevid, med lokalizacijo in navigacijo pa za ugotavljanje lokacije uporablja le poglede.

1.4 Sorodne raziskave

Rešitve, ki se nanašajo na mobilne robote in uporabo numeričnih podatkov iz senzorjev za gradnjo predstavitve prostora in njeno uporabo, so že precej raziskane. V novejšem času je vse več rešitev tudi za vizualno učenje in navigacijo. V nadaljevanju bomo navedli nekaj sorodnih rešitev drugih avtorjev.

Pri obravnavi mobilnih robotov za kompleksnejše naloge hranimo različne podatke, ki omogočajo delovanje sistema in upravljanje z njim. Običajno je sistem razdeljen na več ločenih nivojev, ki so med seboj odvisni. Najnižji nivoji, torej tisti, ki delajo neposredno z napravami na robotu, uporabljajo številske podatke in predstavitve. Višji nivoji pa delajo s kvalitativnimi predstavitvami. Te osnove je Kuipers [17] s sodelavci formaliziral s hierarhijo prostorske semantike (ang. *Spatial Semantic Hierarchy*, SSH). Nivoji, ki jih je uvedel, vključujejo zaznavni, nadzorni, vzročni, topološki in metrični nivo. Ta semantika naj bi bila uporabna tudi pri navigaciji po prostorih širših obsežnosti.

Prednost takšne hierarhije je ta, da lahko na zaznavnem nivoju uporabljamo različne vrste senzorjev. Večinoma so to široko uveljavljeni klasični senzorji, ki merijo razdalje v diskretnih smereh. Primerni pa so tudi panoramski senzorji, za katere so avtorji [24] pokazali, da se skladajo s smernicami SSH.

Pojem vizualnega navigiranja zajema širok spekter postopkov in algoritmov. Možnosti za obdelavo posamezne slike ali zaporedja slik se odsevajo v različnih rešitvah. Najbolj prevladujejo rešitve, ki na slikah iščejo značilke, področja zanimanja ali tokove gibanja slikovnih elementov, in iz njih sklepajo na oddaljenost do predmetov v prostoru [31]. Obsežen pregled teh rešitev opisuje [6]. Večina takšnih rešitev uporablja za primerjavo med trenutnim in nekim znanim pogledom strukturo podatkov, ki je na višjem nivoju od osnovnih intenzitetnih vrednosti.

Panoramske kamere omogočajo panoramski stereo. Tudi če ima robot nameščeno samo eno kamero, lahko dosežemo stereo vid tako, da zajamemo sliki na dveh sosednih lokacijah. Potem moramo na vsaki sliki poiskati značilke, nato pa določiti korespondence na zaporednih slikah. Iz premikov robota in razdalje med korespondencami potem dobimo približek razdalje do značilke [2]. Če pa se robot premika samo po ravnih tleh, lahko predpostavimo, da deli slik, ki se ne premikajo v skladu z ravnino tal, pripadajo predmetom, ki so na ali nad tlemi, in torej predstavljajo oviro. Informacije o globini, ki jih dobimo iz sterea kamer, nam lahko dopolnjujejo tiste podatke, ki jih dobimo iz aktivnih senzorjev [21].

Drugi avtorji pa skušajo dele slike segmentirati in se naučiti medsebojnih prostorskih odnosov teh segmentov. Primer takšnega postopka je segmentacija glede na kvantizacijo barvnih histogramov [28]. Medsebojna lega segmentiranih območij v času navigacije potem omogoča sklepanje na trenutno lokacijo in smer proti cilju.

Tematiki, ki jo obravnavamo v tej nalogi, so bližje tiste rešitve, ki za razpoznavo in sklepanje uporabljajo nizko-nivojske slikovne podatke. Pri teh rešitvah potrebujemo zaporedje slik, ki smo ga zajeli v učni fazi, v testni fazi pa primerjamo trenutno sliko s tistimi iz zaporedja [22, 23]. Vendar pa so predstavljene rešitve veljale za vožnjo po hodnikih, kar je v osnovi enodimenzionalni problemski prostor.

Precej bolj splošna rešitev pa je takšna, ki omogoča gibanje v poljubnem prostoru. Pri razvoju takšnih algoritmov so pomagala tudi dognanja s področja obnašanja žuželk [5], katerih oči so v osnovi panoramski senzorji. Raziskave so pokazale, da si žuželke v bližnji okolici svojega bivališča v spomin vtisnejo razpored izstopajočih delov prizora (okolice). Ko pa se znajdejo dovolj blizu bivališča, jih odmiki med trenutnimi in vtisnjenimi slikami usmerjajo proti cilju.

Postopke, podobne tistim, ki jih uporabljajo žuželke, so v praksi prikazali Mallot, Franz in sodelavci [20, 7]. Uporabili so panoramsko kamero, iz vsake slike pa izkoristili le ozek pas intenzitetnih vrednosti, ki predstavljajo obzorje. Njihovo učenje in lokalizacija temeljita na primerjanju značilk, ki se pojavijo na tem pasu. Podobnost med značilkami na naučeni sliki in na trenutni sliki pove, v katerem območju

prostora se robot nahaja. Smer, ki povzroči povečanje podobnosti značilk, pa jim predstavlja smer proti izbranemu cilju. Prikazali so izgradnjo topološkega zemljevida v obliki grafa. Zemljevid gradijo med raziskovanjem tako, da skušajo hkrati prostor čim bolj pokriti, po drugi strani pa želijo zmanjšati gostoto takih vozlišč grafa, ki bi podali dvoumno lokalizacijo.

Rešitve, ki bi za lokalizacijo in navigacijo uporabljale poglede same, pa so precej redke. Obstoječe metode običajno preslikajo vsako sliko glede na njen izgled ali globalne lastnosti v nek nizko dimenzijski prostor. Primer globalne lastnosti slike je njen barvni histogram [4]. Lahko pa uporabimo navzkrižno korelacijo med delom zabeležene slike ter enako velikim delom trenutne slike [1]. S spremljanjem dveh zaporednih vrednosti korelacije lahko izvajamo sledenje preprostim trajektorijam, ki smo jih prevozili v učni fazi.

Sliko lahko predstavimo tudi z njenimi začetnimi koeficienti Fourierove vrste [12]. Razdalje med koeficienti lahko uporabimo kot mero podobnosti, pri čemer naj bi veljala predpostavka, da je na krajših razdaljah podobnost linearna. Z eksperimenti so Ishiguro in sodelavci pokazali, da lahko prostor, ki ima obliko mreže, predstavimo z modelom vzmeti, ki povezujejo sosedne lokacije. Silo posamezne vzmeti določila stopnja podobnosti, končni model prostora pa predstavlja tak graf, v katerem so sile vzmeti v medsebojnem ravnovesju.

Metoda glavnih komponent sama po sebi ne omogoča ugotavljanja prostorskega konteksta ali medsebojnih prostorskih odnosov elementov na slikah. To lahko delno odpravimo tako, da sliko razdelimo na enakomerne navpične pasove, ki jih obravnavamo ločeno. S tem dobimo informacijo o sosednosti delov izvirne slike v vodoravni smeri. Paletta in sodelavci [27] so to tehniko izkoristili tako, da so s pomočjo Bayesovega sklepanja ugotavljali lokacijo robota.

Pristop, ki uporablja poglede, pa je možno kombinirati s pristopom, ki na sliki poišče značilke. Winters in sodelavci [33] so s panoramskim senzorjem beležili zaporedje slik med vožnjo robota po hodnikih znotraj stavbe. Slike so shranili z metodo glavnih komponent, poleg tega pa so na vsaki od slik iskali daljice, ki sovpadajo z robovi med tlemi in zidovi. Med učenjem so tako zgradili topološki zemljevid, po katerem so potem vizualno navigirali.

Podatke, ki nam jih pripravi metoda glavnih komponent, lahko uporabljamo tudi za verjetnostno računanje trenutne lokacije. Kröse je s sodelavci [16] prikazal delovanje vizualnega učenja goste mreže panoramskih slik z znanimi lokacijami v prostoru z metodo glavnih komponent, ki zgradi verjetnostni model prostora. Lokalizacijo so potem izvajali z iskanjem največje verjetnosti lokacije pogleda v prostoru, verjetnost

lokacije pa so popravljali med premikanjem robota. Njihov postopek je torej deloval po principu predikcije in korekcije.

1.5 Naš pristop

Za vizualno učenje in razpoznavanje se je uveljavila metoda glavnih komponent. Metoda v visokodimenzionalnem prostoru določi nizkodimenzionalni podprostor, s katerim lahko na kompakten način in z največjo stopnjo rekonstrukcije opišemo množico slik. Dobljeni podprostor predstavlja model vhodnih slik, ki je zmožen predstaviti tudi vse tiste slike, ki so vhodnim slikam dovolj podobne. Predstavitve samih slik shranimo kot točke v tem podprostoru. Osnovni postopek se zelo dobro obnese pri množicah slik manjše do srednje velikosti. Večje množice pa presegajo okvire razpoložljivega delovnega pomnilnika, zato postane metoda takrat v praksi neizvedljiva. Metoda namreč najprej med izračunom potrebuje celotno množico podatkov v delovnem pomnilniku. To težavo so kasneje odpravili s postopno izgradnjo podprostora [9] in z združevanjem podprostorov [10].

S postopno gradnjo podprostorov metoda glavnih komponent ni več vezana na zaključeno število vhodnih slik, ampak omogoča dopolnjevanje podprostora in s tem dograjevanje znanja z novimi slikami. Ta lastnost je nujna, če hočemo uporabljati vizualno učenje na robotu. Mobilni robot mora namreč biti sposoben v poljubnem trenutku dopolniti svoje znanje z novimi slikami.

Postopna gradnja je elegantno zmanjšala obsežnost podatkov, ki ga postopek potrebuje med izračuni modela prostora. Problem, ki ga ni rešila, pa se nanaša na količino pomnilnika, ki ga porabimo tudi po izračunih za slike in njihove predstavitve. Izvirne slike moramo namreč hraniti, dokler ni model zgrajen v celoti, šele nato lahko določimo opise slik v tem modelu. Ker ne potrebujemo dvojne predstavitve slik, ki za nameček občutno poveča zahteve po pomnilniku, smo uvedli postopek, ki obravnava predstavitve slik skupaj s postopno gradnjo modela [3].

Izbrano metodo vizualnega učenja smo uporabili za sprotno obravnavo slik, ki jih v različnih intervalih zajema vizualni senzor na mobilnem robotu med raziskovanjem okolja. Model, ki ga metoda izgradi, nam skupaj s predstavitvami slik podaja izgled med učenjem obiskanih lokacij, ki ima obliko topološko-metričnega zemljevida. Tega izkoriščamo, ko z vizualno razpoznavo izvajamo ugotavljanje trenutne lokacije oz. lokalizacijo. Rezultat lokalizacije je ena izmed učnih lokacij, katere izgled je najbolj podoben izgledu robotove trenutne lokacije.

Navigacijo začnemo tako, da uporabnik poda eno izmed učnih slik za želeni cilj

navigacije. Robot nato vizualno določi svojo trenutno lokacijo in se po ravni poti odpravi v smeri, v kateri je ciljna lokacija glede na zemljevid. Na poti periodično vizualno preverja svoj napredek in izvaja morebitne popravke smeri. Navigacijo zaključi, ko z lokalizacijo ugotovi, da je prišel na cilj.

Z izbiro postopka, ki deluje na principu primerjanja pogledov, smo se izognili težavam, ki so prisotne pri postopkih iskanja in sledenja značilk med slikami. Učenje in navigacija s pomočjo pogledov ni vezana na vrsto prostora in strukturo predmetov v njem, ampak potrebuje le dovolj razgibanosti okolja, da lahko iz izgledov sklepa na lokacijo.

1.6 Struktura naloge

V uvodnem poglavju smo uvedli problem, predstavili osnovne pojme, s katerimi operiramo v nadaljevanju, ter navedli motivacijo za raziskave. Podali smo tudi kratek pregled sorodnih del drugih avtorjev.

V naslednjem poglavju bomo predstavili vizualno učenje s pomočjo metode glavnih komponent, njene lastnosti in motivacijo za njeno uporabo. Nato bomo predstavili postopno metodo, ki omogoča postopno gradnjo in dopolnjevanje znanja. Predstavili bomo svojo metodo obravnavanja predstavitev slik, ki za svoje delovanje ne potrebuje preteklih izvirnih slik v pomnilniku.

V tretjem poglavju bomo opisali uporabo vizualnega učenja in razpoznavanja v nalogi učenja prostora in lokalizacije. Predstavili bomo specifične lastnosti vizualnega učenja mobilnega robota ter postopke. Nato bomo opisali postopek za osnovno navigacijo z uporabo lokalizacije.

Četrto poglavje je namenjeno opisom poskusov na laboratorijskem robotu, rezultatom in ugotovitvam. Predstavili bomo poskuse, ki prikazujejo delovanje vizualnega učenja, natančnost lokalizacije ob uporabi različnih postopkov vizualnega učenja ter uspešnost vizualne navigacije.

Nalogo bomo zaključili s petim poglavjem, kjer bomo na kratko povzeli vsebino naloge, poudarili prispevke naloge in podali možne izboljšave in nadaljnje smeri razvoja.

Poglavje 2

Vizualno učenje in razpoznavanje

2.1 Uvod

Učenje je proces, ki podatke uredi na smiseln način, jim pripiše pomen in jih shrani v obliki znanja tako, da lahko kasneje hitro dostopamo do njega. Vizualno učenje kot svoj vir uporablja slike, posnetke oseb, predmetov in okolja v diskretnem času ali v časovnem zaporedju.

Slike prikazujejo številne elemente prizora, ki so lahko zanimivi sami zase ali v nekem medsebojnem odnosu. Slika kot celota pa nam podaja izgled opazovanega prizora. S spreminjanjem zornega kota na prizor ali prizora samega neposredno spreminjamo tudi izgled prizora. Če prizor spreminjamo na sistematičen način, lahko dobimo zvezo med izgledom in parametri prizora. Ta zveza nam potem omogoča povratno sklepanje na parametre prizora iz izgleda samega. Parametre prizora tako razpoznavamo iz izgleda.

Ko z računalniškim sistemom zajamemo sliko prizora, dobimo matriko intenzitetnih vrednosti slikovnih elementov. S časom število slik narašča, in če bi želeli hraniti vse slike, zajete med učenjem, bi s tem občutno obremenili pomnilnik. Shranjevanje slik v njihovi polni obliki in velikosti je nepraktična tudi zato, ker je primerjanje parov slik časovno potratno.

Želimo torej najti tak algoritem, ki na svojem vhodu sprejema slike, iz njih izgradi model, ki predstavlja znanje kot rezultat učenja, in na kompakten način shrani njihove predstavitve. Če slikam pripišemo enolične pomene, želimo model in predstavitve uporabiti za razpoznavanje iz novih slik. Izkaže se, da je eden najprimernejših postopkov izgradnja nizkodimenzionalne predstavitve množice vhodnih slik s pomočjo metode glavnih komponent [25].

2.2 Metoda glavnih komponent

2.2.1 Prostori in podprostori

Slika je v pomnilniku shranjena kot matrika številskih vrednosti na nekem intervalu. Če je slika sivinska, je posamezen slikovni element shranjen kot vrednost, ki predstavlja stopnjo intenzitete. Barvne slike so sestavljene iz več barvnih kanalov, običajno iz treh, zato ima vsak slikovni element po tri vrednosti. Ker meje intervala ne spremenijo informacije, ki jo nosi slika, bomo v nadaljevanju predpostavili, da so intenzitetne vrednosti elementa iz intervala [0..1], pri čemer vrednost 0 predstavlja črn element, 1 pa bel element. Dvodimenzionalno matriko $h \times w$ lahko nato predstavimo kot slikovni vektor velikosti $m = wh$, če slikovne elemente zložimo po vrsti, npr. po stoplcih.

Vsak vektor dolžine m določa točko v m-dimenzionalnem prostoru. Slikovni vektorji se vsi nahajajo znotraj $[0..1]^m$, pri čemer je izhodišče prostora v točki, ki ustreza povsem črni sliki. Množica n slikovnih vektorjev $\mathbf{x}_i \in \mathbb{R}^m$, $i = 1 \ldots n$, torej določa oblak točk v tem visokodimenzionalnem prostoru.

V m-dimenzionalnem prostoru si lahko izberemo nabor m ortonormalnih vektorjev $\mathbf{u}_i \in \mathbb{R}^m$, $i = 1 \ldots m$, ki nam določajo nov koordinatni sistem, ki je glede na izhodiščnega premaknjen in zavrten. Slikovni vektor $\mathbf{x} = [\, x_1,\, x_2,\, \ldots,\, x_m \,]^\top$ lahko v tem koordinatnem sistemu predstavimo kot projekcijo $\mathbf{a} = [\, a_1,\, a_2,\, \ldots,\, a_m \,]^\top$. Če bazne vektorje zložimo v matriko $\mathbf{U} = [\, \mathbf{u}_1,\, \mathbf{u}_2,\, \ldots,\, \mathbf{u}_m \,]$, velja zveza

$$\mathbf{a} = \mathbf{U}^\top (\mathbf{x} - \mathbf{o})\,, \tag{2.1}$$

pri čemer je $\mathbf{o}$ vektor do izhodišča novega koordinatnega sistema. Vektor $\mathbf{a}$ je pri tem *projekcija* vektorja $\mathbf{x}$ v podprostor z baznimi vektorji $\mathbf{U}$ in izhodiščem v $\mathbf{o}$.

Če imamo nabor n slik $\mathbf{x}_i$, $i = 1 \ldots n$, si običajno izberemo takšen podprostor, ki ga razpenja k baznih vektorjev $\mathbf{u}_j$. V tem primeru je smiselni izbor izhodišča $\mathbf{o}$ srednja vrednost vseh vektorjev $\mathbf{x}_i$:

$$\bar{\mathbf{x}} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i\,. \tag{2.2}$$

Nov koordinatni sistem ima potem svoje izhodišče v središču vseh točk. Določiti je treba še, kako bo koordinatni sistem obrnjen. To določimo z izborom ustreznih baznih vektorjev $\mathbf{u}_i$. Bazni vektorji so med seboj ortogonalni in imajo dolžino 1. Če v (2.1) namesto $\mathbf{o}$ uporabimo $\bar{\mathbf{x}}$, dobi projekcija poljubnega vektorja $\mathbf{x}$ obliko

$$\mathbf{a} = \mathbf{U}^\top(\mathbf{x} - \bar{\mathbf{x}})\,. \tag{2.3}$$

Obratna preslikava, imenovana tudi *rekonstrukcija*, ima potem obliko

$$\tilde{\mathbf{x}} = \mathbf{U}\mathbf{a} + \bar{\mathbf{x}} = \sum_{i=1}^{k} a_i \mathbf{u}_i + \bar{\mathbf{x}}\,. \tag{2.4}$$

Enakost $\tilde{\mathbf{x}} = \mathbf{x}$ velja pod pogojem, da prostor med vektorji $\mathbf{U}$ vsebuje točko $\mathbf{x}$. V nasprotnem primeru pa projekcija (2.3) povzroči premik točke v referenčnem prostoru. *Vektor ostanka* (residual) $\mathbf{r}$ predstavlja razliko med obema točkama

$$\mathbf{r} = \mathbf{x} - \tilde{\mathbf{x}}\,. \tag{2.5}$$

Dolžina vektorja $\mathbf{r}$ je *rekonstrukcijska napaka* $r = \|\mathbf{r}\|$.

2.2.2 Iskanje optimalnega podprostora

V praksi se je za izbor baznih vektorjev $\mathbf{u}_i$ zelo dobro obnesla *analiza glavnih komponent* (ang. *Principal Component Analysis*, PCA) [11, 29]. Ta transformacija za $\mathbf{u}_1$ izbere tisto smer, ki ima največjo varianco vhodnih podatkov. $\mathbf{u}_2$ kaže v smer, ki je pravokotna na $\mathbf{u}_1$ in ima največjo varianco, na enak način pa izbere tudi preostale $\mathbf{u}_i$.

Če ima metoda na vhodu n slik $\mathbf{x}_i$, $i = 1 \ldots n$, potem dobimo n ortogonalnih vektorjev $\mathbf{u}_j$, $j = 1 \ldots n$, ki v celoti opišejo vse vhodne slike. Če za vsak vektor $\mathbf{x}_i$, $i = 1 \ldots n$, izračunamo njegovo projekcijo $\mathbf{a}_i$, $i = 1 \ldots n$, z obrazcem (2.3), nato pa za $k = n$ izračunamo rekonstrukcije $\tilde{\mathbf{x}}_i$ s pomočjo (2.4), dobimo $\tilde{\mathbf{x}}_i = \mathbf{x}_i$, $i = 1 \ldots n$.

Za $k < n$ pa takšen rezultat ne velja, in vektor ostanka lahko izračunamo kot

$$\mathbf{r}_i = \mathbf{x}_i - \tilde{\mathbf{x}}_i = \sum_{j=1}^{n} a_{ij}\mathbf{u}_j - \sum_{j=1}^{k} a_{ij}\mathbf{u}_j = \sum_{j=k+1}^{n} a_{ij}\mathbf{u}_j\,. \tag{2.6}$$

Rekonstrukcijsko napako projekcij vhodnih vektorjev lahko torej izrazimo kot

$$r_i = \|\mathbf{r}_i\| = \sqrt{\sum_{j=k+1}^{n} a_{ij}^2}\,. \tag{2.7}$$

Desna stran izraza (2.7) izhaja iz (2.6) in ortonormalnosti $\mathbf{U}$.

Lastnost, ki jo ima podprostor, zgrajen z metodo lastnih komponent, je zagotovljeno minimalna rekonstrukcijska napaka (2.7) za vsak vektor iz množice točk $\mathbf{x}_i$,

$i = 1 \ldots n$, v smislu vsote kvadratov napake za vsak k; noben drug izbor vektorjev $\mathbf{u}_j$ ne omogoča nižje napake.

Izračun baznih vektorjev poteka tako, da najprej izračunamo kovariančno matriko $\mathbf{C} \in \mathbb{R}^{m \times m}$ vhodnih vektorjev

$$\mathbf{C} = \frac{1}{n} \sum_{i=1}^{n} (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^\top = \frac{1}{n} \hat{\mathbf{X}} \hat{\mathbf{X}}^\top , \tag{2.8}$$

pri čemer je $\hat{\mathbf{X}} \in \mathbb{R}^{m \times n}$ matrika, katere stolpec i, $i = 1 \ldots n$, predstavlja vektor $\mathbf{x}_i - \bar{\mathbf{x}}$. Nato rešimo problem lastnih vrednosti, zastavljen kot

$$\mathbf{C}\,\mathbf{U} = \mathbf{U}\,\mathbf{\Lambda}\,, \tag{2.9}$$

pri čemer je $\mathbf{U} = [\,\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n\,]$ matrika *lastnih vektorjev*, $\mathbf{\Lambda}$ pa diagonalna matrika, ki na svoji diagonali vsebuje *lastne vrednosti*, $\boldsymbol{\lambda} = \mathrm{diag}(\mathbf{\Lambda})$. Matrika lastnih vektorjev $\mathbf{U}$ je ortonormalna, zato so lastni vektorji $\mathbf{u}_j$ hkrati tudi bazni vektorji podprostora, ki smo jih iskali. Lastne vrednosti $\boldsymbol{\lambda} = [\,\lambda_1, \lambda_2, \ldots, \lambda_n\,]$ odražajo varianco ustreznih komponent projekcij v smeri pripadajočega lastnega vektorja, zato v nadaljevanju predpostavljamo razvrstitev lastnih vektorjev tako, da velja $\lambda_i \geq \lambda_j$, $1 \leq i < j \leq n$. Število dobljenih lastnih vektorjev in lastnih vrednosti je enako rangu matrike $\mathbf{C}$, ki je enako n, če si vhodne slike med seboj niso linearno odvisne.

Vhodni podatki v naši aplikaciji so slikovni vektorji. Posledica je običajno razmeroma velika vrednost m. Računanje kovariančne matrike (2.8) ter nato razcepa na lastne vektorje in lastne vrednosti (2.9) je zato računsko zahtevna operacija. Če velja $n \ll m$, raje uporabimo *notranji produkt* $\mathbf{C}' \in \mathbb{R}^{n \times n}$ (ki ga delimo z n):

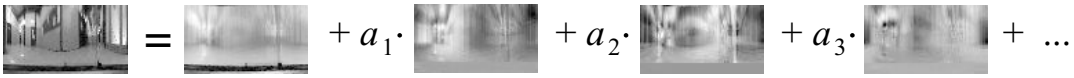
$$\mathbf{C}' = \frac{1}{n} \hat{\mathbf{X}}^\top \hat{\mathbf{X}}\,. \tag{2.10}$$

Matriki $\mathbf{C}'$ pripadajo lastni vektorji $\mathbf{u}'_i$ in lastne vrednosti $\boldsymbol{\lambda}'$. Iz njih dobimo vektorje, ki bi bili rezultat izraza (2.9), z naslednjim izračunom:

$$\lambda_i = \lambda'_i\,, \tag{2.11}$$

$$\mathbf{u}_i = \frac{\hat{\mathbf{X}}\mathbf{u}'_i}{\sqrt{n\lambda'_i}}\,. \tag{2.12}$$

Lastne vektorje in lastne vrednosti dobimo lahko tudi z razcepom singularnih vrednosti (ang. *Singular Value Decomposition*, SVD). Ta postopek se je izkazal kot numerično stabilnejši. SVD razcepi matriko $\mathbf{C}$ na zmnožek treh matrik



Slika 2.1: Linearna kombinacija lastnih slik.

$$\mathbf{C} = \mathbf{U}\,\mathbf{D}\,\mathbf{V}^{\top}. \tag{2.13}$$

$\mathbf{U}$ in $\mathbf{V}$ sta ortogonalni matriki, $\mathbf{D}$ pa je diagonalna matrika singularnih vrednosti. Matriko $\mathbf{U}$ lahko neposredno uporabimo kot matriko lastnih vektorjev, lastne vrednosti pa dobimo kot $\boldsymbol{\lambda} = \mathrm{diag}(\mathbf{D})$.

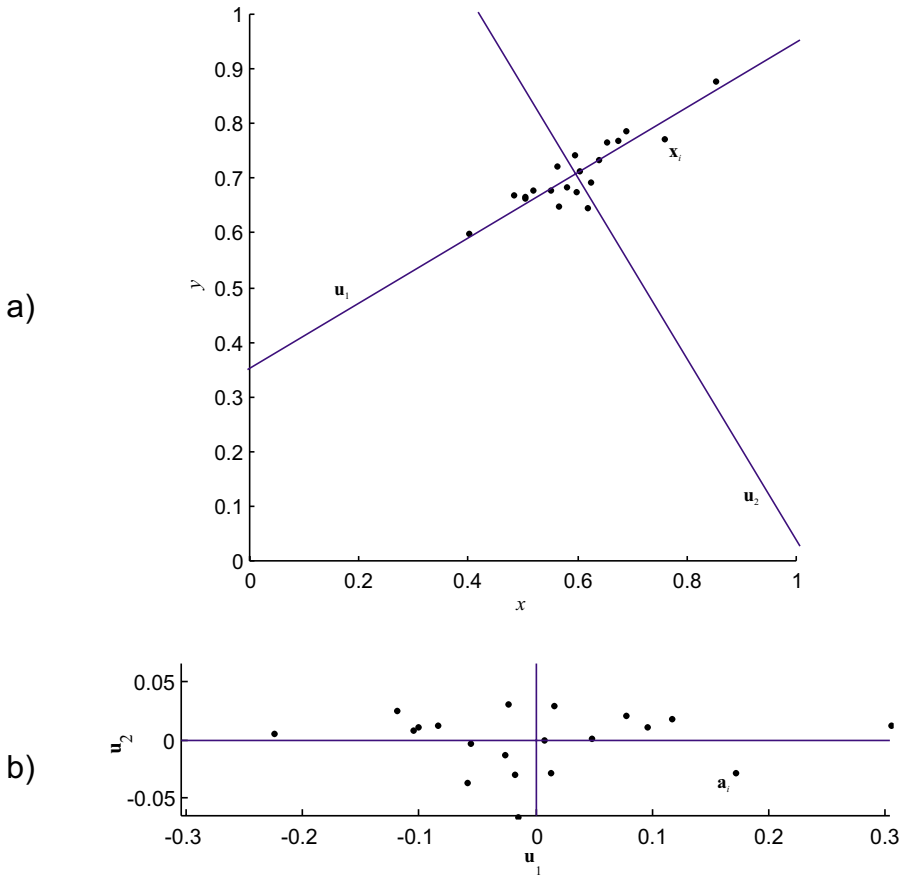
Lastne vektorje, ki jih metoda glavnih komponent zgradi iz množice slik, imenujemo *lastne slike*. Rekonstrukcijo slike dobimo z izrazom (2.4), ki predstavlja linearno kombinacijo lastnih slik. Koeficienti linearne kombinacije so pri tem pripadajoče komponente vektorja projekcije. Na sliki 2.1 vidimo linearno kombinacijo primera slike. Leva stran enačbe predstavlja sliko/rekonstrukcijo. Prvi člen desne strani je povprečna slika množice slik, ki ji pripada rekonstruirana slika. Preostali členi so s pripadajočimi koeficienti pomnožene lastne slike.

2.2.3 Kompaktna predstavitev

Z izračunom lastnih vektorjev kovariančne matrike (2.8) dobimo podprostor, ki vsebuje vse vhodne slike $\mathbf{x}_i$, $i = 1 \ldots n$. To pomeni, da lahko projekcijo $\mathbf{a}_i$ (2.3) vsakega $\mathbf{x}_i$ z izrazom (2.4) popolnoma rekonstruiramo — velikost vektorja ostanka $\mathbf{r}_i$ (2.6) je enaka 0. Slika 2.2 prikazuje primer množice točk v 2D in podprostora, kot ga izbere metoda lastnih komponent. Predstavitev z vektorji $\mathbf{u}_j$ in $\mathbf{a}_i$ torej popolnoma nadomesti izvirno predstavitev $\mathbf{x}_i$. Vendar pa takšna predstavitev zahteva več prostora v pomnilniku, saj sta matriki $\mathbf{X}$ in $\mathbf{U}$ enako veliki, potrebujemo pa še matriko $\mathbf{A} = [\,\mathbf{a}_1,\, \mathbf{a}_2,\, \ldots,\, \mathbf{a}_n\,] \in \mathbb{R}^{n \times n}$.

Količino pomnilnika, ki jo zaseda predstavitev vhodnih slik, zmanjšamo tako, da zmanjšamo število dimenzij podprostora iz prvotnih n na $k < n$. S tem zavržemo del informacije, zato izbor dimenzij, ki jih obdržimo, določimo tako, da so izgube najmanjše. Matoda lastnih komponent nam to izbiro olajša, saj nam zagotavlja, da bo za dane $\mathbf{x}_i$ napaka v smislu vsote kvadratov najmanjša izmed vseh podprostorov velikosti k.

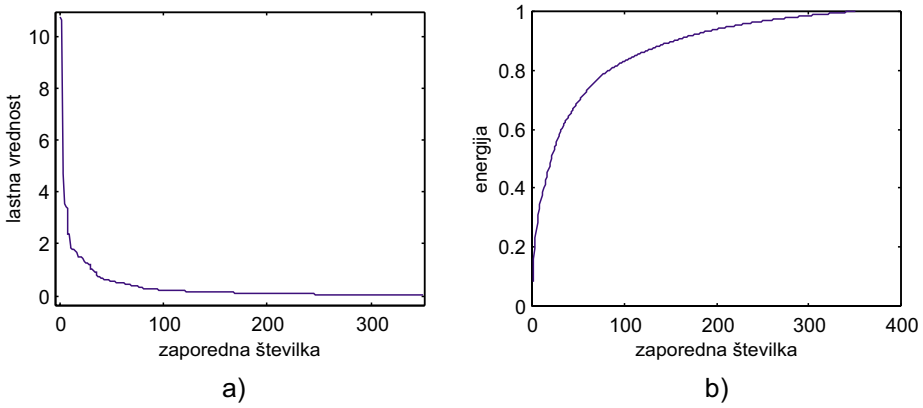
Ker lastne vrednosti λ_j predstavljajo varianco komponent podatkov v smeri pripadajočih lastnih vektorjev $\mathbf{u}_j$, se odločimo za skrajšanje podprostora na prvih k



Slika 2.2: Referenčni prostor, ki vsebuje točke in podprostor (a), in podprostor s točkami, postavljenimi relativno glede na podprosor (b).

dimenzij. Dobimo delno predstavitev vhodnih podatkov, ki jih sedaj sestavljajo projekcije $\mathbf{A} = [\,\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n\,] \in \mathbb{R}^{k\times n}$, pri katerih velja $\mathbf{a}_i = [\,a_{1i}, a_{2i}, \ldots, a_{ki}\,]^\top$, v prostoru lastnih vektorjev $\mathbf{U} \in \mathbb{R}^{m\times k}$ in s pripadajočimi lastnimi vrednostmi $\boldsymbol{\lambda} \in \mathbb{R}^k$. Iz *visoko dimenzionalne* predstavitve slik $\mathbf{x} \in \mathbb{R}^m$ smo tako dobili *nizko dimenzionalno* predstavitev $\mathbf{a} \in \mathbb{R}^k$, pri čemer običajno velja $k \ll m$.

Vrednost k določimo glede na to, kolikšen del informacije smo pripravljeni žrtvovati za manjšo porabo prostora. Če bomo rekonstrukcije slik kasneje prikazovali uporabniku, potem mora biti k nekoliko višji. Za razpoznavo pa je lahko $k < 0{,}1n$. Preprostega pravila za izbiro te vrednosti ni, saj je rezultat precej odvisen od tega, kakšne in kako raznolike so slike na vhodu. Slika 2.3 a) prikazuje značilno krivuljo lastnih vrednosti, ki ima pri višjih indeksih izrazit položni del. Običjano zavržemo lastne vektorje iz tega položnejšega dela, ker je tam varianca najmanjša. Raznolikost slik pa se odraža tudi v porazdelitvi lastnih vrednosti preko lastnih vektorjev, imenovani



Slika 2.3: Značilna krivulja lastnih vrednosti (a) in energije lastnega spektra (b).

lastni spekter, ki jo izrazimo z energijo lastnega spektra E_k:

$$E_k = \frac{\sum_{j=1}^{k} \lambda_j}{\sum_{j=1}^{n} \lambda_j} \,. \tag{2.14}$$

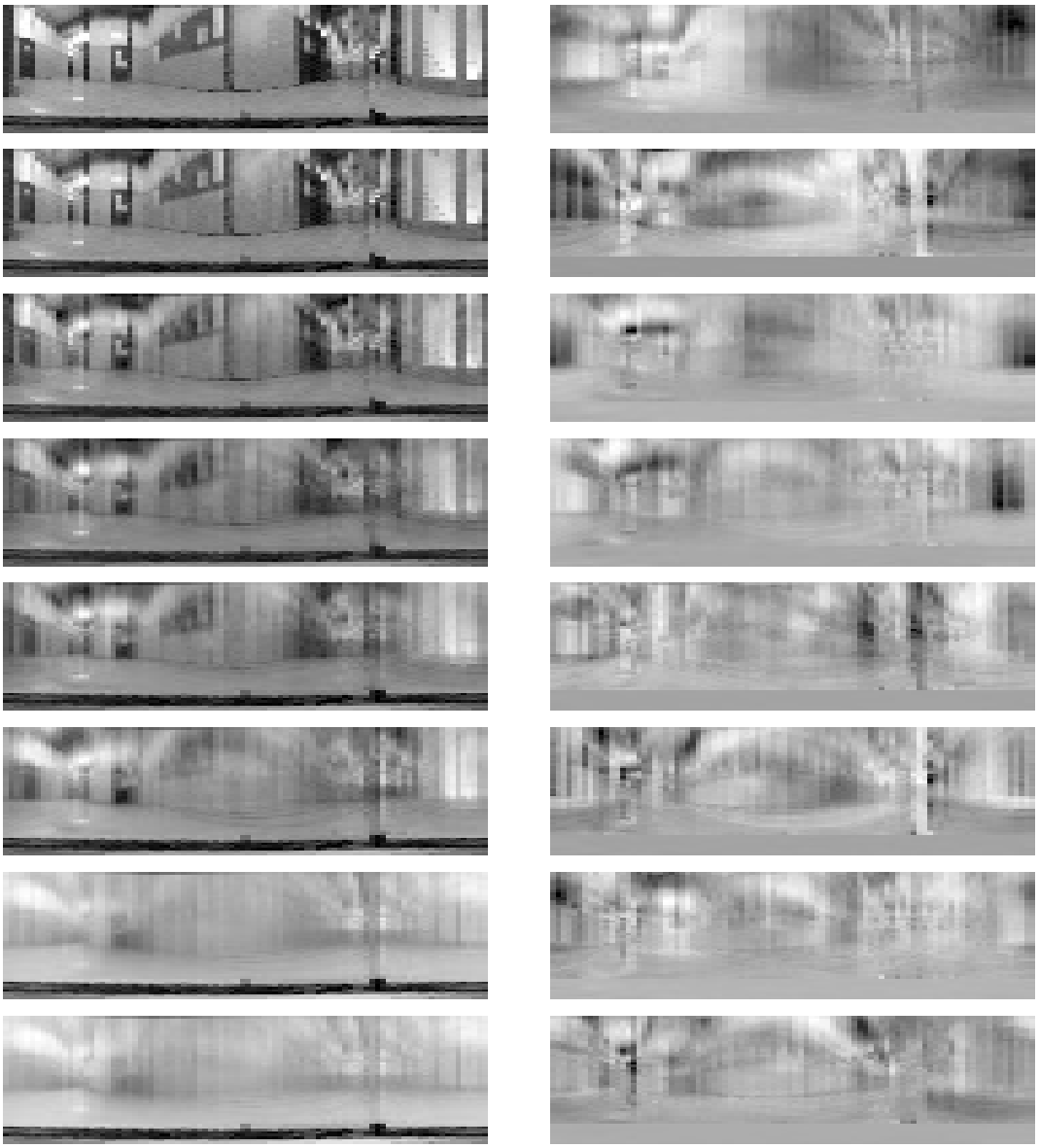
Običajno si izberemo nek delež lastnega spektra τ, ki ga želimo ohraniti tudi po tem, ko zmanjšamo število dimenzij. V tem primeru izberemo $k = \arg\min_j \{E_j \,|\, E_j \geq \tau\}$. Slika 2.3 b) prikazuje krivuljo energije lastnega spektra za lastne vrednosti iz slike 2.3 a).

Vidne učinke zmanjšanja vrednosti k prikazuje slika 2.4 a). Na vrhu je izvirna slika oz. rekonstrukcija pri $k = n = 82$. Z zmanjševanjem vrednosti k pada stopnja vidnih podrobnosti na sliki. Vendar pa tudi pri razmeroma majhnih vrednosti k razločimo posamezne podrobnosti izvirne slike. Slika 2.4 b) prikazuje prvih 8 lastnih slik. Te lastne slike so nosilke najbolj razločujočih značilnosti vhodnih slik.

2.2.4 Primerjanje slik in razpoznava

Če želimo slike uporabljati za razpoznavo, potrebujemo mero za podobnost, na podlagi katere se lahko odločimo, katera od primerjanih slik je najbolj podobna podani sliki. Primer ugotavljanja stopnje podobnosti podaja slika 2.5, kjer vidimo, kako se slika panoramskega senzorja spreminja z razdaljo. Na tem mestu govorimo o podobnosti na nivoju izgleda, ki večinoma sovpada z numerično bližino istoležnih slikovnih elementov. Pogosto uporabljeni meri za takšno podobnost sta navzkrižna korelacija in njena različica v obliki vsote kvadratov razlik (ang. *sum of squared differences*, SSD) [32].

Če imamo nabor slik predstavljen v nekem modelu, je zaželeno, da model omogoča



Slika 2.4: Rekonstrukcije primera ene izmed 82 panoramskih slik hodnika (a) in prvih 8 izmed 82 lastnih slik (b). Rekonstrukcije (a) sestavlja linearna kombinacija prvih k lastnih slik, pri čemer si od zgoraj navzdol sledijo vrednosti $k = 82, 64, 32, 16, 8, 4, 2, 1$.

preprosto medsebojno primerjanje slik. Model, ki ga zgradi metoda glavnih komponent, slike predstavi kot točke v podprostoru, pri čemer so si projekcije podobnih slik bližje skupaj kot manj podobne slike. Takšno obnašanje sovpada s principom podobnosti kot numerične bližine istoležnih slikovnih elementov, ki se odraža kot bližina tako v visoko dimenzionalnem prostoru kot v njegovi nizko dimenzionalni predstavitvi. Zato lahko za mero podobnosti uporabimo evklidsko razdaljo

$$d(\mathbf{x}_i, \mathbf{x}_j) \simeq d(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j) = d(\mathbf{a}_i, \mathbf{a}_j) = \|\mathbf{a}_i - \mathbf{a}_j\|\,, \tag{2.15}$$

pri čemer nižja vrednost $d(\mathbf{x}_i, \mathbf{x}_j)$ pomeni večjo podobnost med $\mathbf{x}_i$ in $\mathbf{x}_j$.

To mero pa lahko povežemo s prej omenjenima merama podobnosti. SSD namreč izračunamo kot

$$c_{\mathrm{SSD}}(\mathbf{x}_i, \mathbf{x}_j) = -\sum_{l=1}^{m} (x_{li} - x_{lj})^2 = -\|\mathbf{x}_i - \mathbf{x}_j\|^2 = -d(\mathbf{x}_i, \mathbf{x}_j)^2\,. \tag{2.16}$$

Če pa vsako vhodno sliko normiramo, tako da za vsak i velja $\|\mathbf{x}_i\| = 1$, potem velja

$$c_{\mathrm{SSD}}(\mathbf{x}_i, \mathbf{x}_j) = -(\mathbf{x}_i - \mathbf{x}_j)^\top (\mathbf{x}_i - \mathbf{x}_j) = 2\mathbf{x}_i^\top \mathbf{x}_j - 2 \tag{2.17}$$

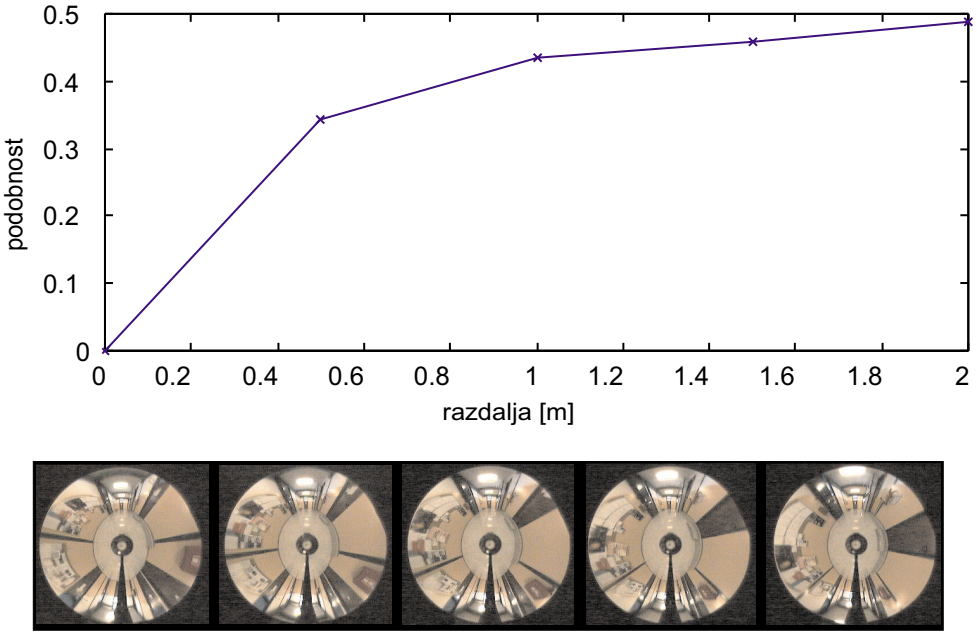
in je torej navzkrižna korelacija $c(\mathbf{x}_i, \mathbf{x}_j)$ med $\mathbf{x}_i$ in $\mathbf{x}_j$

$$c(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^\top \mathbf{x}_j = \frac{1}{2} c_{\mathrm{SSD}}(\mathbf{x}_i, \mathbf{x}_j) + 1\,. \tag{2.18}$$

Primerjava slik z metodo glavnih komponent je torej računsko razmeroma preprosta, saj lahko uporabimo evklidsko razdaljo v nizko dimenzionalnem prostoru. Tudi nove slike, ki niso identične nobeni od učnih slik, vendar so jim dovolj podobne, lahko primerjamo med seboj ali z učnimi slikami tako, da jih projiciramo v podprostor in izračunamo evklidske razdalje.

Razpoznavo lahko potem izvajamo neposredno z mero za podobnost. Običajno namreč učnim slikam dodelimo dodatne oznake, ki na nek način poimenujejo vsebino slike ter podajo dodatne okoliščine, v katerih so slike nastale (npr. položaj kamere ali predmeta, čas dneva ali osvetlitev). Te oznake so namreč med učenjem znane. Pri razpoznavi pa sliko, katere oznake ne poznamo, projiciramo v podprostor in poiščemo najbližnjega soseda glede na mero podobnosti. Z oznako tega soseda potem identificiramo neznano sliko.

Neznana slika pa ne predstavlja vedno nečesa, kar je skupno učnim slikam. Stopnjo združljivosti z učnimi slikami lahko potem ugotavljamo z razdaljo slike od pod-



Slika 2.5: Podobnost slik v odvisnosti od razdalje med lokacijami.

prostora. Najbolj primerna mera za to je velikost vektorja ostanka oz. rekonstrukcijska napaka slike. Če namreč uporabljamo več različnih modelov ali združujemo slike v različne podprostore [19], potem želimo neznano sliko najprej klasificirati oz. izbrati najbolj primeren model. To naredimo tako, da izračunamo razdalje do vsakega od podprostorov, nato pa izberemo tistega, ki je najbližji dani sliki.

2.3 Postopen izračun in dopolnjevanje podprostorov

2.3.1 Uvod

Metoda lastnih komponent je odgovor na nekatere od zahtev, ki smo jih zastavili sistemu. Uspešno reši problem kompaktnejše predstavitve vhodnih slik ob minimalnih izgubah slikovnih informacij. Izgrajena predstavitev omogoča tudi enostavno primerjanje slik, tako tistih, ki smo jih uporabili ob izgradnji podprostora, kot drugih, ki jih projiciramo naknadno.

Izrazi (2.8), (2.10) in (2.9) pa nakazujejo, da je ta postopek namenjen enkratni uporabi za dano vhodno množico slik. Vhodni podatki so kot paket, ki je obdelan naenkrat in v celoti. Če bi želeli podprostor spremeniti tako, da bo upošteval še

nove slike, moramo te dodati k paketu slik in ponoviti izgradnjo.

Uporaba takšne *paketne* metode glavnih komponent na mobilni platformi je nepraktična, saj postopek dodajanja nove slike k predstavitvi ne upošteva trenutno že izgrajenega podprostora. Izračune je tako treba izvesti znova od začetka, ne da bi mogli uporabiti rezultate predhodnih izračunov. Matrika, ki jo izračunamo z izrazom (2.10) in nato razcepimo (2.9), ima velikost $\min(m^2, n^2)$ ne glede na to, da smo prej s kompresijo parameter velikosti zmanjšali na k.

Potrebujemo torej postopek, ki zna obstoječi prostor poljubnih dimenzij dopolniti tako, da bo vseboval novo sliko ter imel enake lastnosti, kot če bi ga zgradili znova s paketno metodo.

2.3.2 Dopolnjevanje predstavitve s posamezno sliko

Pri sistemih za učenje iz izgleda slike običajno zajemamo zaporedno, tako da v različnih intervalih dobivamo nove slike. Te slike bi radi kar takoj vkomponirali v obstoječo predstavitev znanja. Posledica tega je, da se zdaj množica vhodnih slik ter predstavitev z lastnimi prostori v času spreminja. Spremembe se dogajajo vsakič, ko dobimo novo sliko. Zato lahko uvedemo diskretni časovni korak i, ki se nanaša na $\mathbf{x}_i$, torej vhodno sliko, ki se je na vhodu pojavila z zaporedno številko i. Spremenljivke, ki se spreminjajo z vsako novo sliko na vhodu, bodo tako v nadaljevanju v času i nosile indeks (i).

Po preteku časa n smo na vhodu dobili slike $\mathbf{x}_i$, $i = 1 \ldots n$. Iz njih smo zgradili lastni prostor, ki ga razpenjajo lastni vektorji $\mathbf{U}_{(n)} \in \mathbb{R}^{m \times k}$, podprostor ima izhodišče v $\bar{\mathbf{x}}_{(n)}$, pripadajoče lastne vrednosti pa so $\boldsymbol{\lambda}_{(n)}$.

Z naslednjim diskretnim korakom, $n+1$, dobimo vhodno sliko $\mathbf{x}_{n+1}$. Če hočemo, da bodo za podprostor, ki je razpet med vektorji $\mathbf{U}_{(n+1)}$, še vedno veljale lastnosti, ki jih ima prostor zgrajen z metodo glavnih komponent, bomo morali premakniti podprostor na novo izhodišče, bazne vektorje pa zavrteti.

Novo izhodišče podprostora bo v srednji vrednosti vektorjev $\mathbf{x}_i$, $i = 1 \ldots n+1$:

$$\bar{\mathbf{x}}_{(n+1)} = \frac{1}{n+1}(n\bar{\mathbf{x}}_{(n)} + \mathbf{x}_{n+1})\,. \tag{2.19}$$

Podobno lahko dopolnimo tudi kovariančno matriko $\mathbf{C}_{(n+1)}$, ne da bi uporabili vektorje $\mathbf{x}_i$, $i = 1 \ldots n$:

$$\mathbf{C}_{(n+1)} = \frac{n}{n+1}\mathbf{C}_{(n)} + \frac{n}{(n+1)^2}(\mathbf{x}_{n+1} - \bar{\mathbf{x}}_{(n)})(\mathbf{x}_{n+1} - \bar{\mathbf{x}}_{(n)})^\top\,. \tag{2.20}$$

Kovariančne matrike pa ne poznamo eksplicitno, ampak hranimo zgolj njene lastne vektorje in lastne vrednosti. Če bi jo potrebovali, bi jo lahko izračunali kot $\mathbf{C}_{(i)} = \mathbf{U}_{(i)}\mathbf{\Lambda}_{(i)}\mathbf{U}_{(i)}^{\top}$. To bi potem vstavili v izraz (2.20), rezultat pa razcepili na nove lastne vektorje in lastne vrednosti. Vendar pa tak postopek ne prinese dovolj prednosti, cepiti pa bi morali matriko velikosti m^2 (oz., v najboljšem primeru, n^2).

K sreči lahko iz izrazov (2.20) in (2.9) izpeljemo krajšo pot, katere računska zahtevnost je precej manjša, hkrati pa pridobimo dodatne informacije, ki so nujno potrebne kasneje, ko preslikamo točke $\mathbf{a}_{i(n)}$ v nov koordinatni sistem. V nadaljevanju bomo povzeli Hallovo metodo [9].

Vektor $\mathbf{x}_{n+1}$ lahko projiciramo v trenutni lastni prostor, razliko med vektorjem in projekcijo pa označimo kot vektor ostanka $\mathbf{r}_{n+1}$:

$$\mathbf{r}_{n+1} = \mathbf{U}_{(n)}\mathbf{U}_{(n)}^{\top}(\mathbf{x}_{n+1} - \bar{\mathbf{x}}_{(n)}) + \bar{\mathbf{x}}_{(n)}\,. \tag{2.21}$$

$\mathbf{r}_{n+1}$ je pravokoten na vse vektorje v $\mathbf{U}_{(n)}$. To pomeni, da lahko $\mathbf{x}_{n+1}$ popolnoma predstavimo z linearno kombinacijo vektorjev $\bar{\mathbf{x}}_{(n)}$, $\mathbf{r}_{n+1}$ in $\mathbf{u}_{i(n)}$, $i = 1 \ldots k$. Torej je smer, v katero kaže $\mathbf{r}_{n+1}$, primerna za novo smer baze podprostora.

Naj bo $\mathbf{r}'_{n+1}$ normirana različica $\mathbf{r}_{n+1}$:

$$\mathbf{r}'_{n+1} = \begin{cases} \frac{\mathbf{r}_{n+1}}{\|\mathbf{r}_{n+1}\|}, & \|\mathbf{r}_{n+1}\| > 0\,, \\ \mathbf{0}_{m\times 1}, & \text{sicer}, \end{cases} \tag{2.22}$$

kjer je $\mathbf{0}_{m\times 1}$ ničelni vektor dolžine m. Nov nabor lastnih vektorjev $\mathbf{U}_{(n+1)} \in \mathbb{R}^{m\times(k+1)}$ potem dobimo kot

$$\mathbf{U}_{(n+1)} = \left[\, \mathbf{U}_{(n)}\, \mathbf{r}_{n+1} \,\right] \mathbf{R}_{(n+1)}\,, \tag{2.23}$$

kjer je $\mathbf{R}_{(n+1)} \in \mathbb{R}^{(k+1)\times(k+1)}$ ortogonalna matrika rotacije, ki je potrebna, da imajo smeri največje variance pripadajoče komponente podatkov in da so razvrščene po padajočem vrstnem redu glede na lastne vrednosti. Matriki $\mathbf{R}_{(n+1)}$ in $\mathbf{\Lambda}_{(n+1)}$ sta še preostali neznanki, ki pa ju dobimo kot rezultat naslednjega problema lastnih vektorjev:

$$\mathbf{D}_{(n+1)}\, \mathbf{R}_{(n+1)} = \mathbf{R}_{(n+1)}\, \mathbf{\Lambda}_{(n+1)}\,. \tag{2.24}$$

Matriko $\mathbf{D}_{(n+1)}$ sestavimo iz vrednosti lastnega prostora (n) ter vektorja $\mathbf{x}_{n+1}$. Najprej pripravimo pomožni spremenljivki

$$\gamma_{n+1} = \mathbf{r'}^{\top}_{n+1}(\mathbf{x}_{n+1} - \bar{\mathbf{x}}_{(n)}), \tag{2.25}$$

$$\mathbf{a}_{n+1\,(n)} = \mathbf{U}^{\top}_{(n)}(\mathbf{x}_{n+1} - \bar{\mathbf{x}}_{(n)}), \tag{2.26}$$

ki ju uporabimo v izračunu matrike $\mathbf{D}_{(n+1)} \in \mathbb{R}^{(k+1)\times(k+1)}$:

$$\mathbf{D}_{(n+1)} = \frac{n}{n+1}\begin{bmatrix} \mathbf{\Lambda}_{(n)} & \mathbf{0}_{k\times 1} \\ \mathbf{0}^{\top}_{k\times 1} & 0 \end{bmatrix} + \frac{n}{(n+1)^2}\begin{bmatrix} \mathbf{a}_{n+1\,(n)}\mathbf{a}^{\top}_{n+1\,(n)} & \gamma_{(n+1)}\mathbf{a}_{n+1\,(n)} \\ \gamma_{(n+1)}\mathbf{a}^{\top}_{n+1\,(n)} & \gamma^2_{(n+1)} \end{bmatrix}. \tag{2.27}$$

Zdaj imamo vse, kar potrebujemo za dopolnitev lastnega prostora z novo sliko. Kratek povzetek postopka bi bil naslednji:

1. V pomnilniku imamo shranjene vrednosti spremenljivk $\mathbf{U}_{(n)}$, $\boldsymbol{\lambda}_{(n)}$, $\bar{\mathbf{x}}_{(n)}$.
2. Zajamemo novo sliko in jo shranimo v slikovni vektor $\mathbf{x}_{n+1}$.
3. Izračunamo projekcijo $\mathbf{a}_{n+1\,(n)}$ (2.26) in smer vektorja ostanka $\mathbf{r}'_{n+1}$ (2.21) in (2.22).
4. Izračunamo pomožno matriko $\mathbf{D}_{(n+1)}$ (2.27).
5. Izračunamo lastne vektorje in lastne vrednosti matrike $\mathbf{D}_{(n+1)}$ (2.24). Lastne vektorje shranimo kor $\mathbf{R}_{(n+1)}$, lastne vrednosti pa kot $\boldsymbol{\lambda}_{(n+1)}$.
6. Izračunamo bazo podprostora $\mathbf{U}_{(n+1)}$ z uporabo (2.23).
7. Izračunamo izhodišče podprostora $\bar{\mathbf{x}}_{(n+1)}$ s pomočjo (2.19).

Če bi podatke iz prvega koraka uporabili za rekonstrukcijo slik, $\tilde{\mathbf{x}}_i = \mathbf{U}_{(n)}\mathbf{a}_{i(n)} + \bar{\mathbf{x}}_{(n)}$ ter pripisali $\tilde{\mathbf{x}}_{n+1} = \mathbf{x}_{n+1}$, nato na podatkih $\tilde{\mathbf{x}}_i$, $i = 1 \ldots n+1$ paketno izračunali izraze (2.2), (2.8) in rešili (2.9), bi dobili identične vrednosti spremenljivk $\mathbf{U}_{(n+1)}$, $\boldsymbol{\lambda}_{(n+1)}$ in $\bar{\mathbf{x}}_{(n+1)}$. To je zelo pomembna ugotovitev glede na to, da opisani postopek v veliki meri zmanjša število računskih operacij v primerjavi s paketnim postopkom.

2.3.3 Preslikava projekcij v nov lastni prostor

V prejšnjem razdelku smo opisali način, kako razširiti, premakniti in zavrteti bazo podprostora tako, da bo z njo možno ob enakih predpostavkah popolnoma opisati tako vse točke, ki jih je podprostor vseboval, kot tudi neko novo točko, ki jo dodamo

v referenčni koordinatni sistem. Vendar pa postopek obravnava samo podprostor, medtem ko za točke v podprostoru ni poskrbljeno. Če algoritem uporabljamo v opisani obliki, imamo dve možnosti za obravnavo koeficientov.

Prva možnost je ta, da slike projiciramo v podprostor takrat, ko je zgrajen do konca. To pomeni, da moramo vse slike hraniti ves čas delovanja sistema. Za to moramo imeti na razpolago dovolj pomnilnika, ki hkrati hrani vse vhodne slike (nm enot), lastne vektorje (sprva nm enot, ko pa zavržemo nekaj baznih vektorjev, pa km enot), srednjo vrednost (m enot), projekcije slik (2.3) v lastnem prostoru (kn enot) ter lastne vrednosti (sprva n, nato pa k enot). Če na primer vzamemo, da je $k+1 = \frac{n}{2}$, ($k+1$ je število ohranjenih lastnih vektorjev skupaj z vektorjem $\bar{\mathbf{x}}$) in velja $k < n$, potem same slike predstavljajo od $\frac{1}{2}$ do $\frac{2}{3}$ vseh podatkov. Z zmanjševanjem vrednosti k se ta delež še dodatno poveča. To pa je velika redundanca, glede na to, da sam lastni prostor zadovoljivo predstavlja te slike.

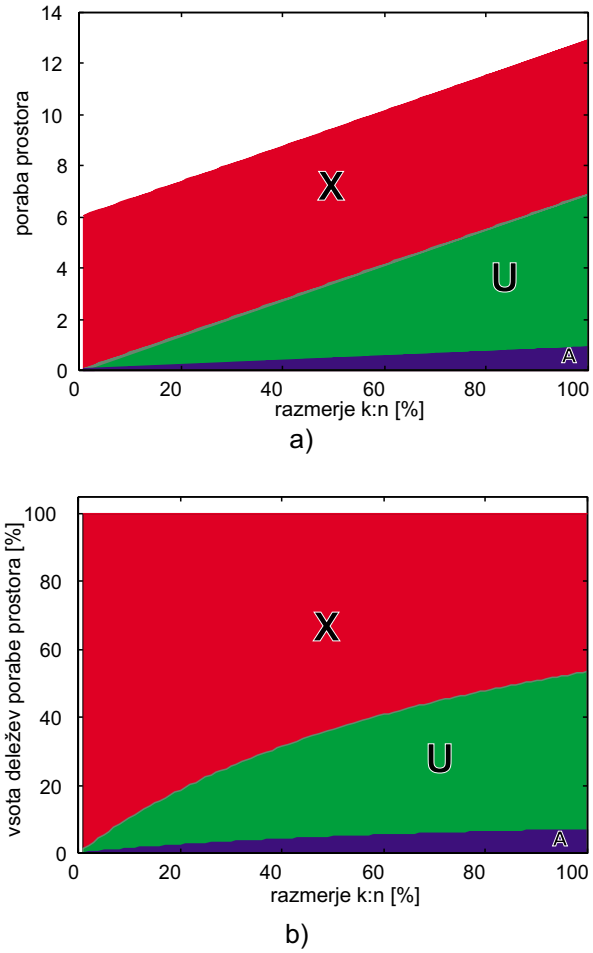
Slika 2.6 še dodatno ilustrira problem porabe pomnilnika. Za ilustracijo smo si izbrali število slik $n = 300$, vsaka slika pa je velika $m = 2000$ elementov. Porabo pomnilnika spremljamo glede na razmerje $\frac{k}{n} \cdot 100\%$. Slika 2.6 a) prikazuje skupno porabo pomnilnika, če v njem hranimo lastne vektorje $\mathbf{U} \in \mathbb{R}^{m\times k}$, matriko projekcij $\mathbf{A} \in \mathbb{R}^{k\times n}$ in matriko vhodnih slik $\mathbf{X} \in \mathbb{R}^{m\times n}$. Vidimo, da slike zasedajo zajeten del pomnilnika. Na sliki 2.6 b) je ta poraba izražena kot delež, ki izmed treh množic podatkov odpade na posamezno množico. Na tej sliki še izraziteje izstopa prevladujoča množica $\mathbf{X}$. Glede na to, da $\mathbf{U}$ in $\mathbf{A}$ skupaj predstavljata približek $\mathbf{X}$, je takšna redundanca precej nesmiselna.

Zaradi smotrne porabe pomnilnika matrike $\mathbf{X}$ med dopolnjevanjem podprostora ne hranimo. Ker pa moramo skupaj z dopolnjevanjem podprostorov ustrezno popravljati tudi projekcije v podprostoru, je druga možnost za obravnavo projekcij slik ta, da projekcije po vrsti rekonstruiramo, rekonstrukcije pa projiciramo v novi podprostor. Z upoštevanjem (2.4) in (2.3) lahko to združimo v

$$\mathbf{a}_{i(n+1)} = \mathbf{U}_{(n+1)}^{\top}(\mathbf{U}_{(n)}\mathbf{a}_{i(n)} + \bar{\mathbf{x}}_{(n)} - \bar{\mathbf{x}}_{(n+1)})\,. \tag{2.28}$$

Za poenostavitev nadaljnjega razmisleka tukaj uvajamo notacijo za opis podprostora. Naj par $(\mathbf{d},\, \mathbf{A}) \in \mathbb{R}^m \times \mathbb{R}^{m\times k}$ predstavlja podprostor, ki ima izhodišče v točki $\mathbf{d}$, razpenjajo pa ga vektorji, ki so zloženi v stolpce matrike $\mathbf{A}$. Podprostor je del referenčnega prostora, torej tistega, ki ga lahko predstavimo s parom $(\mathbf{0}_{m\times 1},\, \mathbf{I}_{m\times m})$. $\mathbf{0}_{m\times 1}$ je pri tem ničelni vektor dolžine m, $\mathbf{I}_{m\times m}$ pa kvadratna matrika identitete velikosti $m \times m$.

Vsak od podprostorov $(\bar{\mathbf{x}}_{(n)},\, \mathbf{U}_{(n)})$ in $(\bar{\mathbf{x}}_{(n+1)},\, \mathbf{U}_{(n+1)})$ je popolnoma vsebovan v



Slika 2.6: Primerjava skupne porabe pomnilnika posameznih elementov (a) in skupni delež porabe (b).

referenčnem prostoru ($\mathbf{0}_{m\times 1}$, $\mathbf{I}_{m\times m}$). Vektorji $\mathbf{x}_i$, $\mathbf{a}_{i(n)}$ in $\mathbf{a}_{i(n+1)}$ za vsak $i = 1 \ldots n$ pa vsi trije predstavljajo isto točko, le da se razlikujejo v tem, glede na kateri podprostor so shranjeni.

Za ilustracijo tega razmisleka prikazuje slika 2.7 primer v dvodimenzionalnem prostoru. Slika 2.7 a) prikazuje dvodimenzionalno množico točk $\mathbf{x}_i$, $i = 1 \ldots n$, v prostoru ($\mathbf{0}_{m\times 1}$, $\mathbf{I}_{m\times m}$), ki jih lahko opišemo s premico. Zato enodimenzionalni podprostor $\mathbf{U}_{(n)}$ v celoti zajema vsako od teh točk. Slika 2.7 b) predstavlja iste podatke v podprostoru ($\bar{\mathbf{x}}_{(n)}$, $\mathbf{U}_{(n)}$), prikazane točke pa so projekcije $\mathbf{a}_{i(n)}$. Ko v referenčni prostor dodamo novo točko $\mathbf{x}_{n+1}$, dopolnimo, premaknemo in zavrtimo podprostor ter dobimo $\mathbf{U}_{(n+1)}$. Slika 2.7 c) prikazuje dopolnitev (modra črtkana črta) prvotnega podprostora ter nov podprostor (rdeči osi). Zaradi nazornosti prikaza

Preslikava	(2.29)	$\mathbf{a}'_{i(n)} \rightarrow \mathbf{a}_{i(n+1)}$
Relativna glede na	$(\mathbf{0}_{m\times 1}, \mathbf{I}_{m\times m})$	$(\bar{\mathbf{x}}_{(n)}, [\,\mathbf{U}_{(n)}\,\mathbf{r}'_{n+1}\,])$
Rotacija	$\mathbf{R}_{(n+1)}$	$\mathbf{R}^{\top}_{(n+1)}$
Translacija	$\Delta\bar{\mathbf{x}}_{(n+1)}$	$-[\,\mathbf{U}_{(n)}\,\mathbf{r}'_{n}\,]^{\top}\Delta\bar{\mathbf{x}}_{(n+1)}$

Tabela 2.1: Povzetek preslikav pri dopolnjevanju podprostorov

$\mathbf{x}_{n+1}$ izrazito odstopa od ostalih točk, česar sicer v praksi običajno ne pričakujemo. Preslikava podprostorov glede na $(\bar{\mathbf{x}}_{(n)}, \mathbf{U}_{(n)})$ je na sliki 2.7 d). Rdeči osi na tej sliki bi sicer dobili, če izvedemo metodo glavnih komponent nad prikazanimi točkami [30]. Slika 2.7 e) pa prikazuje $\mathbf{a}_{i(n+1)}$ glede na $(\bar{\mathbf{x}}_{(n+1)}, \mathbf{U}_{(n+1)})$.

Med $\mathbf{a}_{i(n)}$ in $\mathbf{a}_{i(n+1)}$ obstaja torej pri vseh $i = 1 \ldots n$ preslikava, ki kompenzira preslikavo

$$(\bar{\mathbf{x}}_{(n)}, [\,\mathbf{U}_{(n)}\,\mathbf{r}'_{n+1}\,]) \rightarrow (\bar{\mathbf{x}}_{(n+1)}, \mathbf{U}_{(n+1)})\,. \tag{2.29}$$

Leva stran (2.29) izhaja iz konstrukta (2.23). Na slikah 2.7 c) in d) vidimo preslikavo (2.29) med modrimi in rdečimi osmi, obratno preslikavo, ki deluje nad projekcijami, pa vidimo na prehodu med slikama (2.29) d) in e).

Zaradi združljivosti uvedemo še razširjene vektorje $\mathbf{a}'_{i(n)}$, in sicer kot

$$\mathbf{a}'_{i(n)} = \begin{cases} [\,\mathbf{a}^{\top}_{i(n)}\,0\,]^{\top}\,, & i = 1 \ldots n\,, \\ [\,\mathbf{a}^{\top}_{i(n)}\,\|\mathbf{r}_n\|\,]^{\top}\,, & i = n+1\,. \end{cases} \tag{2.30}$$

Stranski rezultat postopka iz prejšnjega razdelka je ravno podatek o preslikavi (2.29). Ta je sestavljena iz translacije iz izhodišča $\bar{\mathbf{x}}_{(n)}$ v izhodišče $\bar{\mathbf{x}}_{(n+1)}$, torej za vektor $\Delta\bar{\mathbf{x}}_{(n+1)} = \bar{\mathbf{x}}_{(n+1)} - \bar{\mathbf{x}}_{(n)}$, ter rotacije $\mathbf{R}_{(n+1)}$. Ta preslikava je relativna glede na referenčni koordinatni sistem in ne vpliva na točke v njem.

Ko pa želimo preslikati projekcije v podprostoru, moramo upoštevati relativne preslikave glede na ta podprostor. Rotacija točk mora biti obrnjena, torej točke zavrtimo za $\mathbf{R}^{-1}_{(n+1)} = \mathbf{R}^{\top}_{(n+1)}$. Tudi translacija mora potekati v nasprotno smer kot $\Delta\bar{\mathbf{x}}_{(n+1)}$. Hkrati mora biti translacijski vektor znotraj podprostora. Zato za translacijo uporabimo negativno vrednost projekcije tega vektorja v $(\bar{\mathbf{x}}_{(n)}, [\,\mathbf{U}_{(n)}\,\mathbf{r}'_{n+1}\,])$, torej $-[\,\mathbf{U}_{(n)}\,\mathbf{r}'_{n}\,]^{\top}\Delta\bar{\mathbf{x}}_{(n+1)}$. Tabela 2.1 povzema te ugotovitve.

Točke $\mathbf{a}'_{i(n)}$ torej preslikamo v $\mathbf{a}_{i(n+1)}$, $i = 1 \ldots n+1$, z naslednjim obrazcem:

$$\mathbf{a}_{i(n+1)} = \mathbf{R}^{\top}_{(n+1)}(\mathbf{a}'_{i(n)} - [\,\mathbf{U}_{(n)}\,\mathbf{r}'_{n}\,]^{\top}\Delta\bar{\mathbf{x}}_{(n+1)})\,. \tag{2.31}$$

Desno stran izraza (2.31) lahko poenostavimo še naprej:

$$\begin{aligned}\mathbf{a}_{i(n+1)} &= \mathbf{R}^{\top}_{(n+1)}\mathbf{a}'_{i(n)} - \mathbf{R}^{\top}_{(n+1)}[\,\mathbf{U}_{(n)}\,\mathbf{r}'_n\,]^{\top}\Delta\bar{\mathbf{x}}_{(n+1)} = \\ &= \mathbf{R}^{\top}_{(n+1)}\mathbf{a}'_{i(n)} - \mathbf{U}^{\top}_{(n+1)}\Delta\bar{\mathbf{x}}_{(n+1)}\,. \end{aligned} \tag{2.32}$$

Če sedaj označimo preslikavo vektorja $\Delta\bar{\mathbf{x}}_{(n+1)}$ v podprostor $\mathbf{U}_{(n+1)}$ z $\Delta\bar{\mathbf{a}}_{(n+1)}$, dobimo

$$\mathbf{a}_{i(n+1)} = \mathbf{R}^{\top}_{(n+1)}\mathbf{a}'_{i(n)} - \Delta\bar{\mathbf{a}}_{(n+1)}\,,\; i = 1\ldots n+1\,. \tag{2.33}$$

Izraz (2.33) je poenostavitev izraza (2.28), pri čemer namesto matričnega množenja uporabimo rezultat algoritma za dopolnjevanje lastnega prostora.

Naš postopek, ki smo ga opisali v tem razdelku, v celoti nadomesti potrebo po shranjevanju vhodnih slik. S tem smo uvedli precej bolj optimalno uporabo pomnilnika, saj smo sprostili velik del, ki bi ga sicer zasedali statični podatki. Preostanek, ki je potreben za delovanje algoritma, predstavlja dinamične podatke, ki se prilagajajo novim podatkom. Poleg tega smo izkoristili stranske rezultate postopka dopolnjevanja slik za hitrejšo preslikavo projekcij iz starega podprostora (katerega bazne vektorje lahko takoj zavržemo) v novi podprostor.

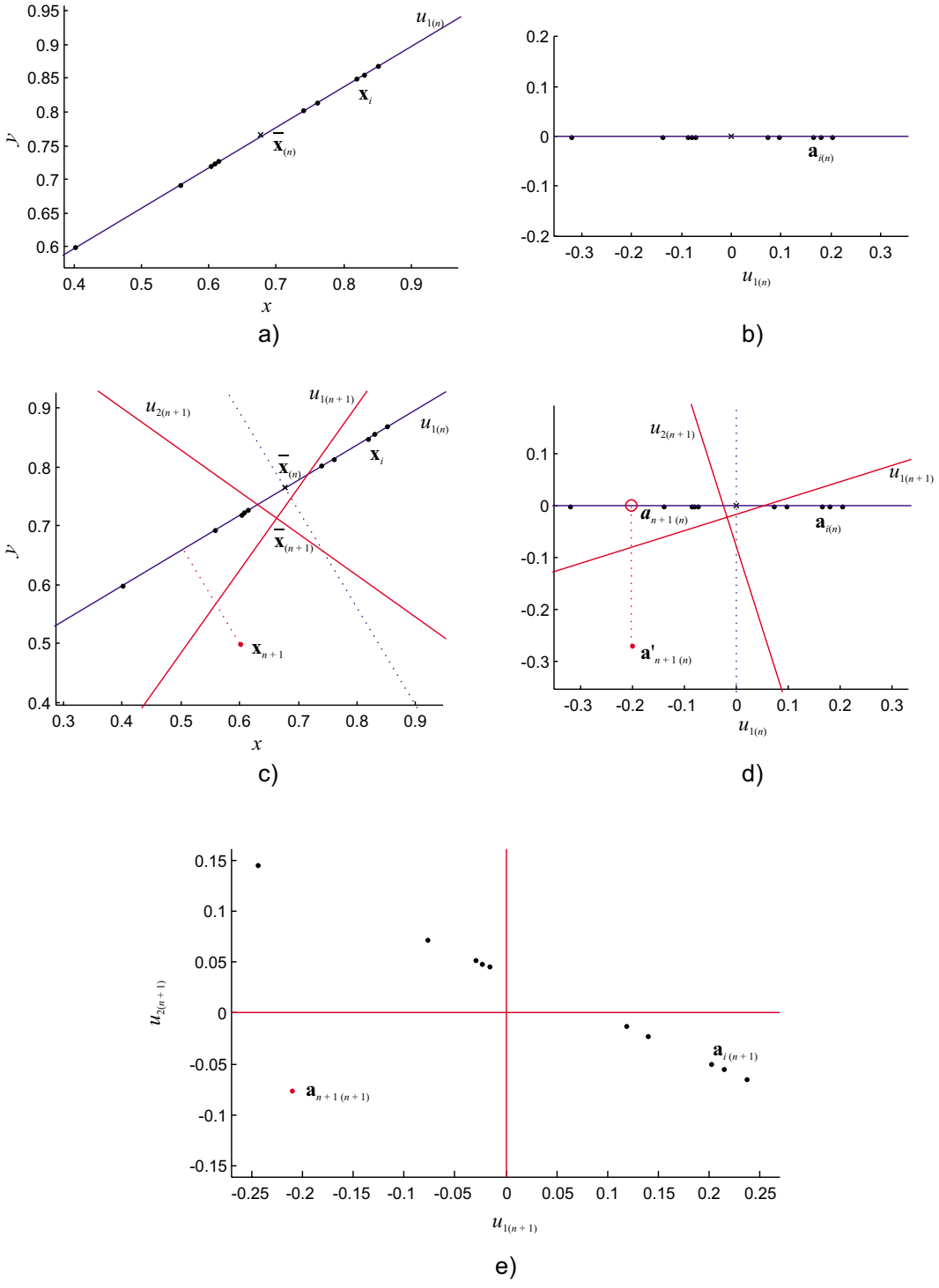
2.3.4 Ohranjanje in povečevanje velikosti podprostora

Paketna metoda glavnih komponent deluje tako, da obdela vseh n vhodnih vektorjev $\mathbf{x}_i$ naenkrat. Rezultat je podprostor velikosti n, ki ga potem zmanjšamo na velikost k, ko zavržemo $n-k$ najmanj pomembnih lastnih vektorjev.

Pri postopni gradnji lastnega prostora pa je pristop nekoliko drugačen, saj lahko v koraku n že imamo lastni prostor velikosti k, kjer je $k < n$. To, kako velik prostor ohranimo do naslednjega koraka, se lahko odločamo sproti.

Ko v lastni prostor velikosti k dodamo novo sliko, bo velikost podprostora narasla na $k+1$. Če takšno linearno rast dopustimo od začetka do konca učenja, bomo na koncu dobili iterativno zgrajen lastni prostor velikosti n, ki bo identičen tistemu, ki ga zgradi paketna metoda. Vendar pa je takšna izgradnja smiselna samo, če želimo izkoristiti popolno sposobnost pomnjenja vhodnih slik, ki jih dobivamo v časovnih razmikih ter sproti brišemo iz spomina.

Prednosti postopne gradnje lastnih prostorov pridejo do izraza, ko je naš cilj zgraditi *delno predstavitev množice slik*, ki bi omogočila ravnotežje med porabo pomnilnika in kvaliteto predstavitve. Poleg tega je lahko, če izberemo $k \ll n$, n pa je zelo velik, za paketno metodo izračun nemogoč, ker preseže količino razpoložljivega



Slika 2.7: Dopolnjevanje podprostorov, prikazano v različnih koordinatnih sistemih. a) in c): referenčni podprostor; b) in d): podprostor $\mathbf{U}_{(n)}$; e): podprostor $\mathbf{U}_{(n+1)}$.

pomnilnika. Rešitev tega problema je ta, da namesto izgradnje celotnega podprostora, ki ga nato zmanjšamo, raje postopno zgradimo manjši podprostor, ne da bi presegli končne velikosti k.

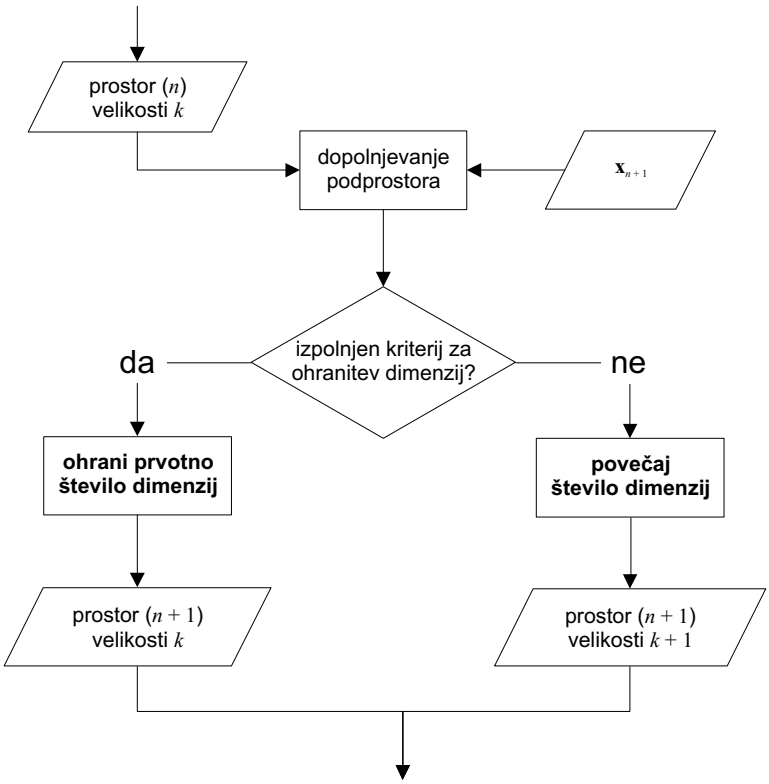
Postopki, opisani v prejšnjih razdelkih, na začetku prejmejo vrednosti spremenljivk $\mathbf{U}_{(n)}$, $\boldsymbol{\lambda}_{(n)}$, $\bar{\mathbf{x}}_{(n)}$ in $\mathbf{a}_{i(n)}$, $i = 1 \ldots n$. Pri tem ni pomembno, na kakšen način so bile te spremenljivke izračunane, veljati morajo samo predpostavke, ki veljajo za rezultat izračuna metode lastnih komponent. Velikost lastnega prostora je $k \leq n$. Iz njega nato dobimo lastni prostor, razpet med $k+1$ lastnih vektorjev, torej podprostor s *povečanim številom dimenzij* (sliki 2.7 b) in e)). Dodatna dimenzija je cena za sposobnost, da rekonstrukcije ostanejo nespremenjene.

Pogosto pa želimo *ohraniti* število dimenzij lastnega prostora k, čeprav smo predstavitev dopolnili z novo sliko. To naredimo tako, da ohranimo le prvih k lastnih vektorjev, torej $\mathbf{U}_{(n+1)} := [\,\mathbf{u}_{1(n+1)}, \mathbf{u}_{2(n+1)}, \ldots \mathbf{u}_{k(n+1)}\,]$. Ustrezno moramo potem popraviti tudi ostale podatke, ki so vezani na velikost lastnega prostora: vsakega od vektorjev $\boldsymbol{\lambda}_{(n+1)}$ in $\mathbf{a}_{i(n+1)}$, $i = 1 \ldots n+1$ skrajšamo tako, da obdržimo le prvih k elementov.

Slika 2.8 prikazuje diagram poteka, ki povzema eno iteracijo dopolnjevanja podprostora, s poudarkom na odločitvi o velikosti podprostora. Proces, označen z „dopolnjevanje podprostora,“ predstavlja postopke, opisane v prejšnjih razdelkih. Sledi preizkušanje kriterija upravičenosti za povečevanje podprostora. Postopek glede na rezultat tega preizkušanja izbere vejo ohranitve velikosti k, ali pa se odloči za povečevanje na $k+1$, ki vrne rezultate, kot jih je vrnil postopek dopolnjevanja podprostora.

Posledica odločitve, da ohranimo število dimenzij, je ta, da podatki niso več predstavljeni v popolni obliki, kot so bili na začetku, ampak da ohranjamo njihove približke. Točke v podprostoru namreč pravokotno projiciramo v smeri zavržene dimenzije, kot to ilustrira slika 2.9. Na tej sliki so točke, ki jih ohranimo, označene z majhnimi krožnicami. Lastnosti metode glavnih komponent nam zagotavljajo, da ob tem naredimo minimalno napako. Vendar pa bi se radi med gradnjo sproti odločali, kdaj število dimenzij ohraniti in kdaj jih povečati.

Kriterije za število zavrženih dimenzij pri paketnih metodah smo že omenili. Te lahko uporabljamo tudi pri sprotni gradnji, vendar nam ne podajo takšne globalne informacije, kot je to v primeru, ko imamo celoten nabor podatkov že podan. Vseeno pa se lahko vnaprej odločimo za neko velikost k, ki ga metoda ne sme preseči. V tem primeru v iteracijah $i \leq k$ vsakič povečamo velikost podprostora, v preostalih iteracijah pa to velikost ohranjamo.

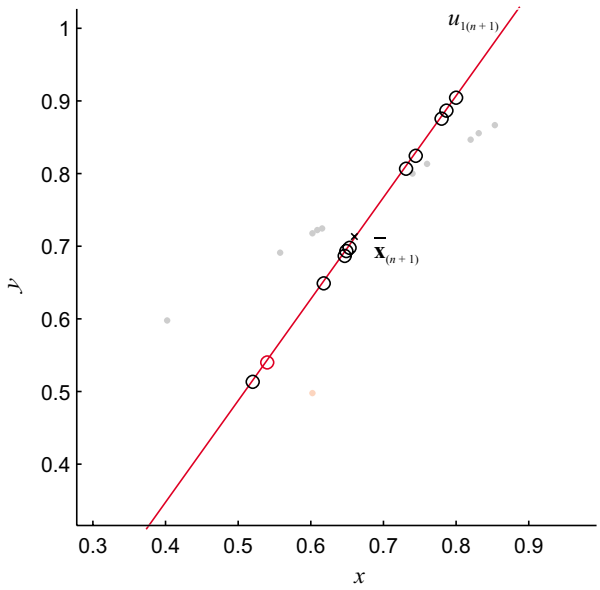


Slika 2.8: Diagram poteka dopolnjevanja podprostora, ohranjevanja in povečevanja podprostora.

Druga možnost je, da si izberemo prag za absolutno ali relativno vrednost $\lambda_{k+1\,(n+1)}$. S tem do neke mere omejimo napako, ki jo naredimo v posameznem koraku. To sicer pomeni, da lahko nadziramo napako le v okviru posamezne iteracije, medtem ko zaradi večdimenzionalnosti obravnavanega prostora in le delne informacije, ki jo ohranjamo, lahko ugotovimo le največjo vrednost napake, ki smo jo do takrat naredili.

Pri določanju kriterija se zato osredotočimo na neposredne učinke, ki jih na podatke povzroči odstranjevanje zadnje dimenzije podprostora. Vrednost napake lahko izračunamo kot vsoto razlik rekonstrukcije slike polnega in skrajšanega lastnega prostora. Za generični lastni prostor z lastnimi vektorji $\mathbf{U} = [\,\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k\,]$, lastnimi vrednostmi $\boldsymbol{\lambda} = [\,\lambda_1, \lambda_2, \ldots, \lambda_k\,]$ in s točkami v tem podprostoru $\mathbf{a}_i = [\,a_{1i}, a_{2i}, \ldots, a_{ni}\,]$ je ta napaka

$$e \;=\; \sum_{i=1}^{n} \left\| \left(\sum_{j=1}^{k} \mathbf{u}_j a_{ij} - \sum_{j=1}^{k-1} \mathbf{u}_j a_{ij} \right) \right\| =$$



Slika 2.9: Prikaz rezultata ohranitve dimenzij podprostora s slike 2.7 a) po dopolnjevanju s točko $\mathbf{x}_{i+1}$ v referenčnem prostoru ($\mathbf{0}_{m\times 1}$, $\mathbf{I}_{m\times m}$).

$$= \sum_{i=1}^{n} \|\mathbf{u}_k a_{ik}\| = \sum_{i=1}^{n} a_{ik}\,, \tag{2.34}$$

pri čemer e predstavlja rekonstrukcijsko napako, ki jo naredimo pri uporabi le prvih $k-1$ lastnih vektorjev $\mathbf{U}$. Poleg nje pa lahko uporabimo tudi srednjo kvadratno rekonstrukcijsko napako (ang. *Mean Squared Reconstruction Error*, MSRE)

$$\begin{aligned} \bar{e}_s &= \frac{1}{n}\sum_{i=1}^{n} \left\| \left(\sum_{j=1}^{k} \mathbf{u}_j a_{ij} - \sum_{j=1}^{k-1} \mathbf{u}_j a_{ij} \right) \right\|^2 = \\ &= \frac{1}{n}\sum_{i=1}^{n} \|\mathbf{u}_k a_{ik}\|^2 = \frac{1}{n}\sum_{i=1}^{n} a_{ik}^2 = \lambda_k\,. \end{aligned} \tag{2.35}$$

Zadnja enakost izraza (2.35) velja, ker lastne vrednosti nosijo vrednosti variance v smeri pripadajočih lastnih vektorjev. Iz računskega stališča je to zelo priročno, saj imamo oceno napake že shranjeno. Vendar pa je ocena $\bar{e}_s$ odvisna od časa i tako, da bo ob konstantnem pragu pri vseh iteracijah dopuščala vedno višjo napako e. Ker pa za vsak pozitven α_i in β_i velja, če je $\sum_i \alpha_i \geq \sum_i \beta_i$, potem velja tudi $\sum_i \alpha_i^2 \geq \sum_i \beta_i^2$, lahko uvedemo oceno vsote kvadratov napake e_s, ki je

$$e_s = \sum_{i=1}^{n} a_{ik}^2 = n\lambda_k\,. \tag{2.36}$$

Poglavje 3

Lokalizacija in osnovna navigacija mobilnega robota

3.1 Uvod

V prejšnjem poglavju smo analizirali postopke za vizualno učenje, ki bi bili primerni za uporabo na mobilnem robotu. V tem poglavju se zato osredotočimo na dejansko uporabo teh postopkov. Vizualno učenje pri tem ne spremeni svoje vloge, saj še vedno predvsem sprejema slike in iz njih gradi vizualno predstavitev prostora. Aplikacija, ki izvaja to učenje, pa mora poskrbeti, da skupaj s predstavitvami slik shrani še podatke o okoliščinah nastanka posamezne slike, torej čas, koordinate in podobne pomembne podatke. Vendar pa ni vseeno, kakšne slike podajamo na vhod vizualnega učenja, ker zlasti orientacija slik zahteva posebno obravnavo. Zato moramo postopek nekoliko prikrojiti.

Vizualno razpoznavanje postane orodje, s katerim sklepamo na trenutno lokacijo iz slikovne informacije in naučenega modela. Preden lahko izvedemo uspešno lokalizacijo, pa moramo slike obravnavati na način, ki je združljiv s postopkom učenja. Znana lokacija robota je nato osnova za izvedbo navigacije do izbranega cilja. Z lokalizacijo namreč izvemo tudi, v katero smer moramo poslati robota, če hočemo, da pride na cilj. Navigacijo izvajamo postopno v korakih, ki robota pripeljejo do bližnjega podcilja, tam pa robot preveri uspešnost napredka navigacije.

V nadaljevanju bomo obravnavali posebnosti aplikacije na mobilnem robotu, predlagali ustrezne postopke za učenje in lokalizacijo ter izvedbo osnovne navigacije, ki jo vodijo vizualne informacije.

3.2 Vizualno učenje z mobilnim robotom

Mobilni robot med vožnjo spremlja vrtenje koles. Akumulacija hitrosti in pospeškov pogonskih motorjev nam sproti daje oceno relativne lokacije robota. Če na njegovi poti zajemamo slike iz kamere, lahko te slike opremimo z oznako lokacije $\mathbf{l}_i = [l_{xi}, l_{yi}, l_{\alpha i}]^\top$. Pri tem predstavljata l_{xi} in l_{yi} odmik v smeri x in y glede na začetno lokacijo, $l_{\alpha i}$ pa predstavlja kót α med začetno in trenutno usmeritvijo robota.

Med raziskovanjem prostora dobimo množico slik $\mathbf{x}_i$, posnetih na pripadajočih lokacijah $\mathbf{l}_i$. Slika ponazarja vizualno informacijo o lokaciji in okolici te lokacije. Učenje nato poteka tako, da to informacijo na smiseln način shranimo skupaj s podatkom o lokaciji in usmeritvi robota. Način učenja in izbor informacije morata biti združljiva s postopki, ki jih potem uporabljamo pri uporabi naučenega znanja.

Naš pristop k vizualnem učenju je takšen, da je informacija, ki jo hranimo, slika sama. Iz nje namreč ne poskušamo izluščiti nobene informacije o dejanski vsebini slike. Ves sistem učenja in, kasneje, razpoznavanja lokacije sloni na podobnosti med slikami. Pri tem izkoriščamo predpostavko, da je podobnost med dvema slikama sorazmerna z razdaljo njunih pripadajočih lokacij.

Takšen pristop ima naslednje prednosti:

- pri učenju in razpoznavi ne uvajamo pričakovanj o izgledu okolice in predmetov na posamezni sliki, kar je sicer običajno treba uvesti, če se odločamo za značilke, ki jim sledimo na slikah,
- slikovno informacijo izkoristimo v celoti, ne osredotočamo se le na manjše dele slike,
- ker slik ne analiziramo, s tem ne uvajamo dodatnih napak in negotovosti, ki so sicer posledica slikovnega šuma, približkov pri računanju in numeričnih napak,
- primerjava z učnimi slikami poteka hitro.

Zgoraj opisane predpostavke in postopki pa imajo tudi nekaj slabosti:

- posamezna slika, shranjena kot vektor, ne pove ničesar o okolici robota, saj *semantike* ali pomena elementov prizora niti ne poskušamo odkriti,
- prostori, ki jih raziščemo, redko ostanejo nespremenjeni skozi čas, večinoma pa se posamezni elementi spreminjajo, robot se srečuje z dinamičnimi elementi (npr. ljudje, ki se gibljejo), spreminja pa se lahko tudi osvetlitev prostora.

Težave, ki sledijo iz omenjenih slabosti, pa lahko razrešimo tako, da uporabimo dovolj temeljito metodo pri raziskovanju, pri razpoznavi pa uvedemo postopke, ki so robustne na spremembe osvetlitve in dodane ali zakrite elemente prizora.

Postopek učenja v grobem poteka takole:

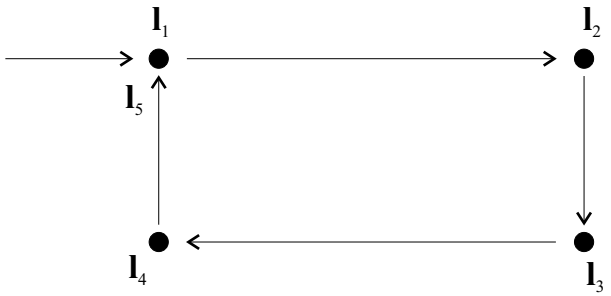
- mobilni robot se vozi po prostoru in na vsaki od lokacij $\mathbf{l}_i$ zajame sliko $\mathbf{x}_i$, $i = 1 \ldots n$,
- zajete slike sproti ali periodično uporabimo kot vhod izgradnje in dopolnjevanja lastnega prostora z lastnimi vektorji $\mathbf{U}_{(n)}$, izhodiščem prostora v povprečju učnih slik $\bar{\mathbf{x}}_{(n)}$ in lastnimi vrednostmi $\boldsymbol{\lambda}_{(n)}$,
- predstavitve slik $\mathbf{x}_i$, $i = 1 \ldots n$, shranimo kot projekcije $\mathbf{a}_{i(n)}$, $i = 1 \ldots n$, v trenutnem lastnem prostoru, pripišemo jim podatke o njihovi lokaciji $\mathbf{l}_i$, $i = 1 \ldots n$, izvirne slike $\mathbf{x}_i$, $i = 1 \ldots n$, pa odstranimo iz pomnilnika.

„Zemljevid,“ ki ga tako dobimo, je kombinacija topološke in metrične predstavitve prostora. Lastnosti metričnih predstavitev se kažejo v tem, da imamo shranjene lokacije v prostoru, izražene z dvema koordinatama v enoti razdalje (npr. v metrih). Topološko predstavitev pa nakazujejo povezave med lokacijami, ki sicer vsaj sprva nakazujejo le časovno zaporedje obiskov lokacij. Vendar če predpostavimo zveznost raziskovanja prostora v smislu, da izključimo možnost vmesne „ugrabitve“ robota, potem te povezave označujejo tudi *dostopnost* sosednih lokacij. Dostopnost se kaže kot dejstvo, da je v času raziskovanja med dvema sosednima lokacijama robot prevozil ravno pot, ne da bi vmes naletel na oviro.

Robot se giblje po tleh na kolesih. Kamera je zato ves čas na konstantni višini od tal, kar nam v osnovi tvori planarni oz. dvorazsežni problemski prostor. Gibanja v smeri pravokotno na tla torej ni. To velja tudi, če dopuščamo, da se robot lahko giblje po spustih, vzpetinah ali stopnicah. Takrat se nam sicer zgodi, da metrični del predstavitve na istem območju predstavlja več kot le en prostor. Vendar pa topološki del poskrbi za konsistentnost podatkov. V skrajnem primeru lahko uvedemo še podatek o „plasti“ ali „nadstropju“, ki bi prišel prav tudi, če dopustimo možnost, da robot na svoji poti uporabi dvigalo.

Vsaka lokacija $\mathbf{l}_i$ pa je izražena s tremi komponentami. Tako kot prvi dve komponenti lahko tudi smer, v katero je robot obrnjen, vpliva na izgled slike $\mathbf{x}_i$. Če je na robotu nameščena običajna perspektivna kamera, usmerjena v smer robotovega gibanja, je lahko pogled iz istega mesta, vendar z različnimi smermi gledanja, povsem drugačen. Če pa robot zaznava izgled okolice s panoramskim senzorjem, pa

dobimo sliki, ki se v grobem razlikujeta le za zamik stolpcev. Kljub temu pa sistem, ki slike primerja na na nivoju istoležnih elementov vektorjev, sam po sebi ne more ugotoviti, da sta si sliki izredno podobni ali celo enaki. Problem lahko ilustriramo z obiskom oglišč pravokotnika med vožnjo, kot jo prikazuje slika 3.1. Na vsakem oglišču zajamemo po eno sliko. Ko pridemo v točko $\mathbf{l}_5$, na diagramu vidimo, da smo se ponovno znašli v točki $\mathbf{l}_1$. Izgled slike iz $\mathbf{l}_5$ se namreč razlikuje od izgleda slike iz $\mathbf{l}_1$. Če uporabljamo klasično perspektivno kamero, bosta sliki popolnoma različni, panoramski sliki pa vsebujeta isto informacijo, zamaknjeno za ustrezno število stolpcev. V naslednjem razdelku bomo opisali, kako ta problem lahko rešujemo.



Slika 3.1: Ilustracija problema pravokotnega potovanja.

3.2.1 Poravnava slik

Problemski prostor robota je v realnosti torej trirazsežen. Pri učenju moramo to dejstvo upoštevati. V osnovi imamo tako dve tehniki, ki ju lahko uporabimo:

- problemski prostor ohranimo trirazsežen, pri učenju imajo vse tri komponente $\mathbf{l}_i$ enakovredno vlogo, ali
- problemski prostor predstavimo kot dvorazsežen, smer pa je samo dodaten podatek, ki fiksira tretjo razsežnost.

Če se odločimo za prvi pristop, moramo upoštevati, da navkljub planarnosti območja gibanja smeri gledanja ne moremo preprosto zanemariti. Vsako mesto si moramo ogledati pri čim več rotacijah. Fizično lahko to izvajamo tako, da robota vrtimo na mestu oz. okrog središčne osi panoramske kamere. Vendar pa je prednost panoramske slike, ki jo hranimo v cilindrični obliki, da lahko z zamiki stolpcev dobimo približke slik, ki bi jih dobili z ustrezo rotacijo kamere okrog osi projekcije. Zamik slike za p stolpcev naredimo tako, da stoplec c, $c = 0 \ldots w-1$, iz izvirne slike

prenesemo na mesto stoplca $(c + p)$ mod w zamaknjene slike. To lahko naredimo, ker se na cilindričnih slikah prizor ovije po modulu (slika 1.3 b).

Simuliranje vrtenja kamere ali robota je preprost način za rešitev problema rotacije. Ob vsaki sliki $\mathbf{x}_i$ postopek učenja dobi na vhod r različic slik $\mathbf{x}_i^{+p}$, kjer naj oznaka $+p$, $p = 0 \ldots r-1$, predstavlja p-ti odmik stolpcev izvirne slike in velja $\mathbf{x}_i^{+p} = \mathbf{x}_i^{+0} = \mathbf{x}_i$. Pri sprotnem učenju podprostora to pomeni, da lastni prostor $\mathbf{U}_{(i-1)}$, $\bar{\mathbf{x}}_{(i-1)}$, $\boldsymbol{\lambda}_{(i-1)}$ in $\mathbf{a}_{(i-1)}^{+p}$ sedaj dopolnimo tako, da upošteva vse slike $\mathbf{x}_i^{+p}$, $p = 1 \ldots r$, ter pripadajoče lokacije $\mathbf{l}_i^{+p} = [\, l_{xi},\, l_{yi},\, (l_{\alpha i} + p\gamma) \bmod 2\pi \,]$. γ je pri tem diskretni kot, ki je določen s širino posameznega stolpca slike: $\gamma = \frac{2\pi}{w}$. Posledično velja, da se število vhodnih podakov pomnoži za faktor r, število vseh vektorjev $\mathbf{a}_i^{+p}$ je torej ri. Ustrezno se podaljša tudi postopek učenja. Vrtenje panoramskih slik ima sicer nekaj specifičnih lastnosti, ki omogočajo hitrejši postopek izračuna podprostorov [13].

Drugi pristop predvideva, da ima problemski prostor le dve prostostni stopnji, tretja pa je neka funkcija prvih dveh ali slike, ki jo na tem mestu zajamemo. Najbolj intuitivna rešitev je, da so vse slike obrnjene enako, da torej za vsak i velja $\mathbf{l}_i = [\, l_{xi},\, l_{yi},\, \alpha \,]^\top$. Takšna rešitev je možna, če imamo dostop do zunanjega ali notranjega kompasa. Notranji kompas je lahko naprava, ki zazna Zemljin magnetizem ali deluje po principu vztrajnosti (vrtilna količina). Za zunanji kompas pa je med gibanjem v zunanjem okolju ob sončnem vremenu moč izkoristiti polarizacijo sončeve svetlobe. Kompas nam poda informacijo o odklonu od „severa“ δ_i, ki je kót, za katerega preuredimo stolpce slike $\mathbf{x}_i$, da kompenziramo ta odklon.

Kompasa pa nimamo vedno na razpolago, zato moramo določiti $\delta_i = g(l_{xi},\, l_{yi})$ ali, še bolje, $\delta_i = f(\mathbf{x}_i)$. Možen izbor za f je postopek, ki poišče na panoramski sliki ničelno fazo [26]. Vendar funkcija vrača različne vrednosti, če je na vhodu prisoten šum.

Mi smo se odločili za učenje uporabiti tisto različico $\mathbf{x}_i^{+p}$ slike $\mathbf{x}_i$, ki je najbližja lastnemu prostoru z izhodiščem v $\bar{\mathbf{x}}_{(i-1)}$ in baznimi vektorji $\mathbf{U}_{(i-1)}$. Iskani p nam vrne naslednji izraz:

$$p = \arg \min_{l=0\ldots r-1} \|\mathbf{U}_{(i-1)}\mathbf{U}_{(i-1)}^\top(\mathbf{x}_i^{+l} - \bar{\mathbf{x}}_{(i-1)}) + \bar{\mathbf{x}}_{(i-1)}\| \tag{3.1}$$

oz. izberemo tisti p, za katerega je rekonstrukcijska napaka projekcije $\mathbf{x}_i^{+p}$ v lastni prostor $(i-1)$ najmanjša. Izbrano sliko $\mathbf{x}_i^{+p}$ nato v koraku i uporabimo na način, opisan v prejšnjem poglavju. Preden pa shranimo pripadajoči $\mathbf{l}_i$, moramo komponento $l_{i\alpha}$ popraviti, da upošteva rotacijo, ki jo naredimo z zamikom stolpcev slike.

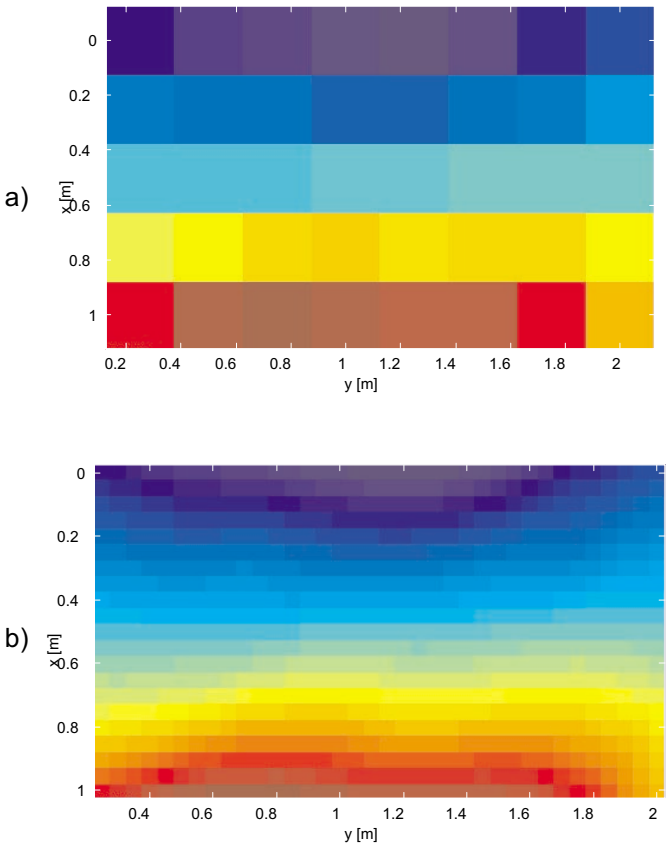
3.2.2 Parametrični prostori

Učenje, opisano v predhodnih razdelkih, nam ustvari razmeroma redko vzorčene podatke o prostoru, ker je znanje predstavljeno kot množica diskretnih pogledov na prostor. Kvaliteta znanja je odvisna tako od konfiguracije prostora kot od gostote vzorčenja. Na osrednjem delu velikega prostora je tako potreben precej večji premik, da se bosta dva pogleda razlikovala v enaki meri, kot se dva bližnja pogleda v majhnem prostoru, polnem predmetov. Podobno pogledi vzdolž dolgega ravnega hodnika, ki na stenah nima vidnih nobenih elementov, med seboj niso bistveno različni. Vendar tudi če predpostavimo, da se robot uči prostora, ki je dovolj razgiban, je malo verjetno, da bo med lokalizacijo in navigacijo obiskal natanko iste lokacije kot med učenjem.

Če je topološko-metrična shema vzpostavljena dovolj dobro, da med sosedi ni prevelikih razmikov, lahko umetno povečamo število vzorcev. To lahko naredimo tako, da opredelimo $\mathbf{a}_i = f_a(l_{xi}, l_{yi})$ oz. $\mathbf{a}_i = f_a(l_{xi}, l_{yi}, l_{\alpha i})$. S tem določimo zvezo med predstavitvami slik in parametri nastanka teh slik. Vrednosti funkcije f_a imamo podane v tistih točkah parametrov, ki smo jih obiskali med raziskovanjem. Funkcija f_a se spreminja zvezno v odvisnosti od njenih parametrov. Z enodimenzionalnim parametrom tako v podprostoru opiše posplošeno krivuljo, če ima parametra dva, predstavlja posplošeno površino itd. Vsaka točka na tej krivulji ali ploskvi določa sliko z enoličnimi parametri. Razpoznava s slik bi tako lahko potekala z najmanjšo razdaljo do krivulje ali površine. Ker pa funkcija ni analitična, točnih vrednosti v neznanih točkah ne moremo izračunati. Lahko pa dobimo vmesne vrednosti z interpolacijo funkcije po parametričnem prostoru (x, y, α).

Z interpolacijo dosežemo približek gostejšega in enakomernega vzorčenja prostora. Ko takšno predstavitev uporabljamo za lokalizacijo, lahko dosežemo točnost, ki je precej večja od tiste, ki jo lahko dobimo z redko vzorčenim prostorom.

Sliki 3.2 in 3.3 prikazujeta primer uporabe interpolacije, kjer je prostor vzorčen s korakom 25 cm. Z interpolacijo smo naredili približek vzorčenja s korakom na 5 cm. Slika 3.2 a) prikazuje vrednosti a_{1i}, torej prve komponente projekcij, porazdeljenih glede na lokacije nastanka $\mathbf{l}_i$ pripadajočih slik $\mathbf{x}_i$. Interpolacija po prostoru (x, y) nam vrne vrednosti, kot jih prikazuje 3.2 b). Izgled točk $\mathbf{a}_i$, če jih prikažemo v prostoru njihovih prvih treh komponent, prikazuje slika 3.3. Točke rdeče barve predstavljajo projekcije slik, modre točke pa so približki, ki jih dobimo z interpolacijo.



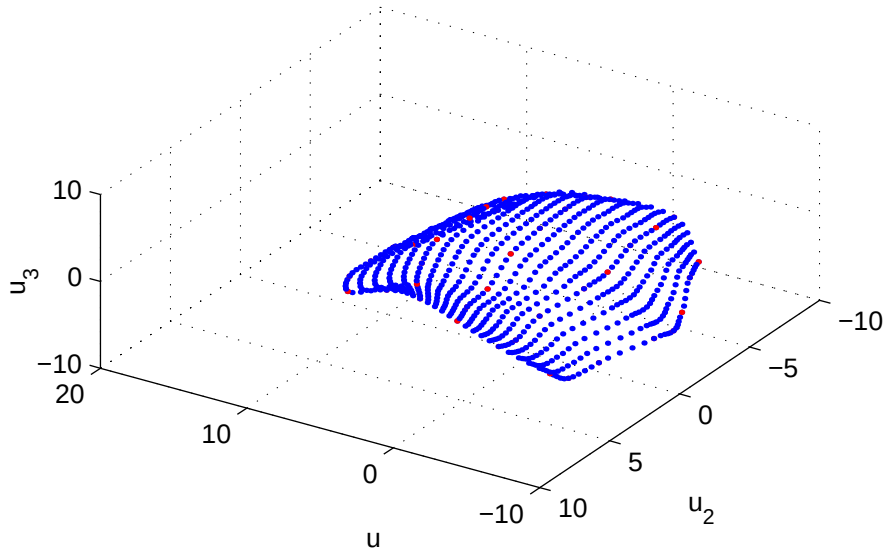
Slika 3.2: Vrednosti a_{1j}, prikazane na pripadajočih lokacijah (l_{xj}, l_{yj}), $j = 1 \ldots n$; običajno vzorčenje (a) in interpolacija (b).

3.3 Lokalizacija

Lokalizacija je postopek, ki iz naučenega znanja in trenutnih zaznav sklepa na lokacijo, s katere so možne trenutne zaznave. V našem primeru uporabljamo zemljevid tako, da na njem poiščemo lokacijo, ki se najbolj sklada s trenutnim izgledom okolice.

Pri lokalizaciji mobilnega robota zemljevid predstavlja znanje, ki ga zberemo z učenjem. Znanje mora biti predstavljeno tako, da je združljivo s podatki, ki jih prejmemo med lokalizacijo. V našem primeru je znanje predstavljeno z modelom slikovne informacije učnih slik ter projekcijami teh slik v modelu. Lokalizacija je nato izvedena kot iskanje znane slike, katere izgled je najbolj podoben sliki, ki jo je robot zajel na trenutni lokaciji. Rezultat lokalizacije je lokacija, ki jo hranimo ob najbližjem sosedu projekcije trenutne slike v lastnem prostoru.

Po obisku n lokacij je rezultat učenja podprostor z izhodiščem v $\bar{\mathbf{x}}_{(n)}$, razpenjajo ga lastni vektorji $\mathbf{U}_{(n)}$, projekcije slik pa hranimo v vektorjih $\mathbf{a}_{i(n)}$. Ob vsaki pro-



Slika 3.3: Prve tri dimenzije podprostora s projekcijami slik (rdeče pike) in interpolacija med njimi glede na parametre (modre pike).

jekciji hranimo tudi lokacijo $\mathbf{l}_i$. Če se robot nahaja nekje na območju, ki smo ga raziskovali med učenjem, potem lahko dobimo približek robotove lokacije tako, da v podprostor projiciramo sliko $\mathbf{y}$, ki jo zajamemo na tej lokaciji:

$$\mathbf{b} = \mathbf{U}_{(n)}^{\top}(\mathbf{y} - \bar{\mathbf{x}}_{(n)})\,, \tag{3.2}$$

nato pa izmed $\mathbf{a}_{i(n)}$, $i = 1 \ldots n$ poiščemo najbližjega točki $\mathbf{b}$:

$$M = \arg\min_{i=1\ldots n} \|\mathbf{b} - \mathbf{a}_{i(n)}\|\,. \tag{3.3}$$

Približek lokacije, na kateri smo zajeli sliko $\mathbf{y}$, je torej $\mathbf{l}_M$.

Tak pristop je možen, če lahko iz pogleda neposredno določimo vse iskane parametre, torej tako lego kot orientacijo. To pa je možno z uporabo kompasa ali s shranjevanjem zadostnega števila zavrtenih slik na vsaki lokaciji. Lokalizacija pri tem v splošnem ne diskriminira nobene od dimenzij problemskega prostora.

Če pa med učenjem slike poravnavamo tako, da shranimo le po eno reprezentativno rotacijo, pa je treba opisani postopek dopolniti. Naš pristop k izbiri različice $\mathbf{x}_i^{+p}$ je bil tak, da smo izbrali tisto sliko, ki je bila trenutno izgrajenemu modelu najbližja. Ta kriterij lahko uporabimo tudi pri lokalizaciji. Iz slike $\mathbf{y}$ pripravimo w simuliranih rotacij $\mathbf{y}^{+l}$, $l = 0 \ldots w-1$, kjer je w širina slike, nato pa po obrazcu

(3.1) dobimo indeks rotacije p. $\mathbf{y}^{+p}$ nato projiciramo z obrazcem (3.2), poiščemo najbližjega soseda (3.3), rezultat lokalizacije pa je potem lokacija

$$\mathbf{l}_y = \begin{bmatrix} l_{xM} \\ l_{yM} \\ \left(l_{\alpha M} - p\frac{2\pi}{w}\right) \bmod 2\pi \end{bmatrix}. \tag{3.4}$$

Kriterij (3.1) je odvisen od modela v času $i-1$. To pomeni, da lahko v času $n > i-1$ z njegovo uporabo dobimo drugačno vrednost za isto sliko $\mathbf{x}_i$. Možna posledica je, da lokalizacija spodleti ali vrne napačen rezultat.

Namesto razdalje do modela pa lahko za izbor najboljše poravnave slike uporabimo razdalje do najbližje projekcije učnih slik $\mathbf{a}_{i(n)}$, $i = 1 \ldots n$. Z vsako izmed w različic slik $\mathbf{y}$ torej izvedemo razpoznavo, izberemo pa tisto različico, ki ima najvišjo stopnjo podobnosti z eno izmed učnih slik. Glede na to, da se robot nahaja v bližini učnih lokacij, je velika verjetnost, da bo učnim slikam najbolj podobna ravno tista slika, ki bo imela želeno poravnavo.

Opisana lokalizacija potem poteka tako, da najprej izračunamo projekcije $\mathbf{b}^{+l}$ slik $\mathbf{y}^{+l}$ z uporabo (3.2). Nato za vsako od projekcij poiščemo najbližjo projekcijo učnih slik

$$M_l = \arg\min_i \left\| \mathbf{b}^{+l} - \mathbf{a}_{i(n)} \right\|, \tag{3.5}$$

pri čemer dobimo tudi najmanjšo razdaljo $d_l = \|\mathbf{b}^{+l} - \mathbf{a}_{M_l(n)}\|$. Nato poiščemo indeks najmanjše d_l

$$p = \arg\min_{l=0\ldots r-1} d_l, \tag{3.6}$$

rezultat lokalizacije pa za $M = M_p$ dobimo iz (3.4).

3.4 Osnovna navigacija

Navigacija je proces, pri katerem skuša robot potovati od svoje začetne pozicije do nekega izbranega cilja. Sistem upravlja s pogonom robota, običajno mu pošilja želeno hitrost in smer gibanja. Učinke teh akcij pa kot povratno informacijo dobiva v obliki notranjih in zunanjih stanj. Notranje stanje dobi v obliki ocene trenutnega položaja in smeri gibanja, zunanje pa iz raznih senzorjev.

Na tem mestu obravnavamo navigacijo v znanem okolju. Okolje je znano v tem smislu, da ga je robot predhodno raziskal, zajel vrednosti zunanjih senzorjev ter jih

v kombinaciji z znanimi notranjimi stanji uporabil za izgradnjo modela okolja. Ta model predstavlja znanje, ki ga nato robot izkorišča med navigacijo.

Navigacija lahko poteka na več nivojih. Višji nivoji navigacije poskrbijo, da robot poišče optimalno pot skozi večje in zapleteno okolje. Sem spada tudi izogibanje oviram, ki v osnovi predstavlja le sprotno izbiro nove poti, če se trenutna izkaže za neprehodno. Na osnovnem nivoju pa skrbi za to, da je ocena trenutne lokacije robota zadovoljiva in da izbere ustrezno smer gibanja za naslednji korak.

Iskanje optimalnih poti in izogibanje oviram sta v umetni inteligenci in robotiki že dobro raziskana problema. Za svoje delovanje potrebujeta le ustrezno topološko ali metrično predstavitev okolja, ki ga dostavijo nižji nivoji navigacije, ter delujoče osnovne korake, ki prevedejo akcije na pogon robota. Zato nas v tem delu zanima predvsem izvedba slednjih, torej osnovnih komponent navigacije.

Lokalizacija, opisana v predhodnem razdelku, je pomemben element pri osnovni navigaciji, saj nam poda informacijo o tem, kje, sodeč po zaznavah senzorjev, se robot nahaja. Če robota aktiviramo na neznani lokaciji v znanem okolju, je to ključen korak, ki omogoči začetek navigacije. Trenutno lokacijo robota poveže s tisto znano lokacijo, ob ali v neposredni bližini katere se nahaja z največjo verjetnostjo. Hkrati izvemo tudi približek smeri, v katero je robot usmerjen, in to ga umesti v znani model prostora. Tako je neposredno možno robota usmeriti do naslednje točke, ki je na poti do izbranega cilja.

Če se izkaže, da je možno iz trenutne lokacije do končnega cilja priti po ravni poti, bi lahko robota napotili neposredno v smeri proti cilju in ga ustavili šele, ko se notranje stanje pokrije s shranjenim notranjim stanjem cilja. Če to ni možno, pa lahko algoritem za iskanje najkrajše poti določi takšno zaporedje vmesnih ciljev, da je vsak vmesni cilj dosegljiv iz predhodnega po ravni poti.

Postopek takšne slepe vožnje do cilja je nepriporočljiv, ker ne upošteva napak, ki so nujno prisotne pri takšnem procesu. Viri napak so naslednji:

- robotovo notranje zaznavanje trenutne lokacije je relativno natančno na krajših razdaljah, pri daljših razdaljah pa običajno zaznamo akumulacijo napake in vse večjega odstopanja od resnične lokacije,
- podatki iz senzorjev ne določijo lokacije dovolj enolično, ker so tudi na drugem delu znanega okolja podali podobne rezultate,
- tudi če je lokalizacija zanesljiva in enolična, nam poda lokacijo, ki ni točno enaka trenutni robotovi lokaciji, ampak je nekje v njegovi neposredni bližini.

Zaradi omenjenih razlogov je tudi trivialne poti priporočljivo razčleniti na krajše korake. Tako dobimo precej manjše napake, ki jih sproti zaznavamo in popravljamo.

V nadaljevanju bomo opisali postopek osnovne navigacije, ki upošteva gornje ugotovitve. Pri tem bomo zaradi enostavnejše ponazoritve zanemarili orientacije, zato bomo lokacije označevali kot točke $\mathbf{q}_i \in \mathbb{R}^2$. Posebej bomo označili ciljno točko $\mathbf{q}_f$. Postopek poteka tako, da najprej izvedemo lokalizacijo, ki nam vrne koordinate lokacije $\mathbf{q}_1$, torej oceno izhodiščne lokacije robota glede na naučeno znanje in trenutne podatke senzorjev. Robota nato premaknemo za vektor $\mathbf{s}_1$ proti cilju. Ko se robot ustavi, ponovimo lokalizacijo, ki vrne $\mathbf{q}_2$. Postopek ponavljamo, dokler nam lokalizacija ne vrne $\mathbf{q}_i = \mathbf{q}_\mathrm{f}$.

Po vsaki lokalizaciji i izračunamo vektor smeri $\boldsymbol{\delta}_i$ med $\mathbf{q}_i$ in $\mathbf{q}_f$:

$$\boldsymbol{\delta}_i = \mathbf{q}_\mathrm{f} - \mathbf{q}_i \,. \tag{3.7}$$

Ker je v večini primerov razdalja do cilja predolga, izberemo le krajši del vektorja $\boldsymbol{\delta}$ v dolžini ρ. Vektor dela poti $\mathbf{s}_i$, za katerega premaknemo robota, nato izračunamo kot

$$\mathbf{s}_i = \begin{cases} \rho \frac{\boldsymbol{\delta}_i}{\|\boldsymbol{\delta}_i\|}, & \|\boldsymbol{\delta}_i\| > \rho \,, \\ \boldsymbol{\delta}_i, & \text{sicer.} \end{cases} \tag{3.8}$$

S tem postopkom omejimo dolžino koraka navigacije, ker tako omogočimo pogosto preverjanje napredka navigacije ter korekcije morebitnih napak.

Pri opisanem postopku smo lokalizacijo na koraku $i > 1$ izvedli na enak način, kot v prvem koraku. To pomeni, da smo vedno poizvedovali po vsem naučenem prostoru, robotu pa pri tem ni bilo potrebno pomniti zgodovine navigacije. Vendar potem pri takem postopku ni pomembno, kje je bilo izhodišče navigacije, koliko korakov je že opravil in kakšno pot je prevozil. To sicer deluje dobro, če je možno vsako lokacijo enolično opisati s podatki iz senzorjev. V nasprotnem primeru pa lahko lokalizacija doživi preskoke, ki povzročijo, da $\mathbf{q}_i + \mathbf{s}_i \not\approx \mathbf{q}_{i+1}$. Poleg tega po nepotrebnem preiskuje vse znane točke, čeprav bi se lahko omejil na majhno podmnožico in pri tem izkoristil zgodovino navigacije.

Optimalnejši postopek bi upošteval dodatno informacijo o opravljeni navigaciji ter izrazil lokalnost okolice trenutne lokacije. Najpreprostejši način za izvedbo takšnega postopka je z uporabo radija λ. Z njim na koraku $i + 1$ omejimo problemski prostor, v katerem lokaliziramo robota, na tiste $\mathbf{l}_j$, za katere velja $\|\mathbf{q}_i + \mathbf{s}_i - [\, l_{xj},\, l_{yj} \,]^\top\| < \lambda$. Radij λ ima sprva dovolj veliko vrednost, da zajame večino znanega prostora, nato pa se zmanjša na nek vnaprej podani minimum λ_min.

Pri realizaciji opisanega postopka navigacije moramo upoštevati, da robota krmilimo v njegovem koordinatnem sistemu, ki ima običajno svoje izhodišče v lokaciji, kjer smo ga vključili. Zemljevid prostora pa je predstavljen v svojem koordinatnem sistemu. Četudi koordinatna sistema poravnamo in poskrbimo, da izhodišči v obeh koordinatnih sistemih sovpadata, ne moremo predpostaviti, da se bosta pokrivala tudi kasneje. Zato moramo te razlike upoštevati tako pri lokalizaciji kot pri izvrševanju ukazov na robotu.

Ko je robot v lokaciji in orientaciji, ki ju njegov koordinatni sistem opiše z $\mathbf{l}'_i$, nam lokalizacija vrne vektor $\mathbf{l}_i$, ki je opisan v koordinatnem sistemu zemljevida. Ker predpostavimo, da je lokalizacija uspela, gre torej za isto lokacijo. Razliko med orientacijama izrazimo kot

$$\beta_i = l'_{i\alpha} - l_{i\alpha}\,, \tag{3.9}$$

kar nam poda kót, za katerega moramo okrog robotove lokacije zavrteti koordinatni sistem, da postaneta oba enako usmerjena. Preslikavo med točkami odometrije robota in lokacijacijami na zemljevidu opisuje homogena preslikava $\mathbf{T}_i$:

$$\mathbf{T}_i = \begin{bmatrix} 1 & 0 & l'_{ix} \\ 0 & 1 & l'_{iy} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos(\beta_i) & -\sin(\beta_i) & 0 \\ \sin(\beta_i) & \cos(\beta_i) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -l_{ix} \\ 0 & 1 & -l_{iy} \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.10}$$

Celoten postopek izvedbe koraka osnovne navigacije je torej naslednji. Ko je robot v lokaciji in orientaciji $\mathbf{l}'_i$, dobimo z lokalizacijo $\mathbf{l}_i$. Iz podatkov iz teh vektorjev izračunamo $\mathbf{T}_i$ (3.10). Nato iz točke $\mathbf{q}_i = [\, l_{ix},\, l_{iy}\,]^\top$ izračunamo vektor naslednjega koraka $\mathbf{s}_i$ (3.8). Robota želimo poslati v točko, ki jo koordinatni sistem zemljevida opiše z vektorjem $\mathbf{g}_{i+1} = \mathbf{q}_i + \mathbf{s}_i$. Ker pa robot operira v svojem koordinatnem sistemu, ga moramo poslati v točko $\mathbf{g}'_{i+1}$ iz njegovega koordinatnega sistema, ki jo dobimo s homogeno preslikavo

$$\begin{bmatrix} \mathbf{g}'_{i+1} \\ 1 \end{bmatrix} = \mathbf{T}_i \begin{bmatrix} \mathbf{g}_{i+1} \\ 1 \end{bmatrix}. \tag{3.11}$$

Poglavje 4

Eksperimenti

4.1 Uvod

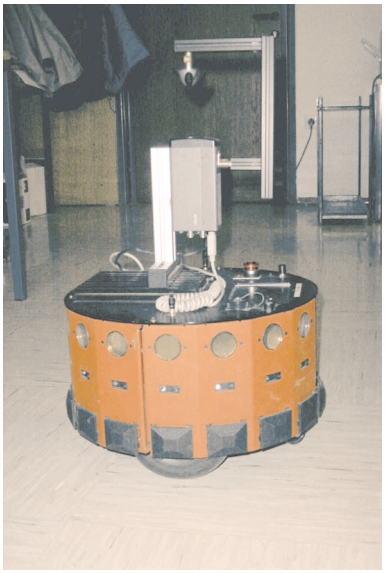
V predhodnih poglavjih smo podali pregled postopkov za vizualno učenje, razpoznavanje, lokalizacijo in navigacijo. V tem poglavju pa želimo predstaviti empirične ugotovitve ter rezultate, ki jih dobimo z opisanimi algoritmi.

Eksperimenti so združeni v tri sklope. V prvem delu bomo preizkusili, kako se obnese učenje s postopnim dodajanjem slik in naknadnim spreminjanjem predstavitve. Drugi del se nanaša na lokalizacijo v naučenem okolju, tretji pa prikazuje uspešnost osnovne navigacije mobilnega robota.

4.1.1 Testna platforma

Algoritme, opisane v predhodnih poglavjih, smo empirično preizkusili z laboratorijskim robotom Magellan Pro (slika 4.1). Komponente robota upravlja vgrajeni računalnik s procesorjem Pentium II, trdim diskom in sistemom Linux. Robot ima dve pogonski kolesi z ločenima pogonoma. Pogon omogoča vožnjo naprej in vzvratno vožnjo poljubne hitrosti, zavijanje ter obračanje na mestu.

Vgrajeni senzorji vključujejo dva tipa globinskih senzorjev ter zaznavanje trka. Globinski senzor je infrardeči in zvočni, vsakega tipa po 14 enakomerno razporejenih enot po obodu. Na robota smo namestili še video kamero, ki je priključena na analogno-digitalni pretvornik računalnika, ki omogoča zajemanje žive digitalne slike. Kamero smo usmerili tako, da je njena projekcijska os usmerjena navpično, pred objektiv pa smo namestili hiperbolično zrcalo. S tem smo dobili katadioptično kamero, ki zajema panoramo v obsegu 360°. Dobljene hiperbolične slike smo sproti razvili v cilindrične slike (slika 1.3) in delali samo s slednjimi.



Slika 4.1: Testna platforma, robot Magellan Pro z nameščenim panoamskim senzorjem.

Programsko je vse sisteme robota mogoče upravljati preko vmesnikov CORBA. Računalnik je lahko povezan v brezvrvično lokalno omrežje po standardu IEEE 802.11, kar omogoča porazdeljeno upravljanje na več robotih ali sodelovanje z namiznimi računalniki.

S pomočjo vmesnikov CORBA smo sprogramirali aplikacijo, ki periodično bere lokacijo robota ter meritve globinskih senzorjev. S pomočjo teh podatkov v pomnilniku vzdržuje verjetnostno mrežo, ki z vrednostmi v celicah pove verjetnost, da tisto celico pokriva kakšen objekt [21]. Aplikacija to mrežo sproti vizualizira skupaj z oznako lokacije in usmeritve robota ter omogoča preprosto vodenje robota s klikanjem po zaslonu.

Poleg interaktivnega dela aplikacija omogoča prototipu testne aplikacije dostop do podatkov o trenutnem stanju robota, ter možnost napotitve robota v izbrano točko. Čeprav potem navigacija poteka v celoti vizualno, nam globinski senzorji omogočajo elegantno preprečevanje trkov robota ob predmete v prostoru. V našem delu sicer ne obravnavamo problematike izogibanja oviram, ne moremo pa izključiti možnosti pojavitve ovire na dani poti.

Slika 4.2: Panoramski sliki hodnika (a) in laboratorija (b) v ločljivosti 50×90, ki jo uporabljamo za učenje in navigacijo.

4.1.2 Potek poskusov

Poskuse smo izvajali v prostorih laboratorija ter na hodniku fakultete. Oba tipa prostorov sta zanimiva, ker na hodnikih tipično prevladujejo velike površine, ki jih predstavljajo prazne stene, in je torej nekoliko manj elementov v prizoru, vendar vrata in oglasne deske omogočajo razlikovanje med posameznimi lokacijami (slika 4.2 a). Notranjost laboratorija pa je tipično založena s pohištvom in opremo, zato je raznolikost velika in diskriminatornost slik večja (slika 4.2 b).

Naprava za zajemanje žive slike nam omogoča neposredno uporabo slik v aplikaciji, ta pa vse slike shranjuje skupaj z ostalimi podatki, kot so čas in odometrija. Tako lahko slikovne podatke uporabljamo sproti ali kasneje, izven neposredne platforme. Čeprav za samo delovanje aplikacije shranjevanje slik ni potrebno, nam je ta možnost prišla prav kot vir slik za poskuse, opisane v naslednjem razdelku.

4.2 Vizualno učenje

Paketna metoda glavnih komponent zagotavlja najmanjšo rekonstrukcijsko napako učnih slik izmed vseh podprostorov velikosti k, $k < n$. Ker ob izgradnji pozna vse vhodne slike, usmeri podprostor tako, da je zanje rezultat najboljši tudi po tem, ko nekaj dimenzij zavržemo. Če gradnjo izvajamo postopno, pa dimenzije krajšamo sproti, torej na način, ki ni optimalen za končni nabor slik. Poleg tega vsakič, ko se odločimo za ohranitev dimenzije, povzročimo perturbacijo predstavitev slik v referenčnem prostoru, saj naredimo pravokotno projekcijo v podprostor, ki točk ne vključuje. Zato pričakujemo slabše rezultate v primerjavi s paketno metodo.

Vendar pa vsakič, ko po dopolnjevanju podprostora ohranimo njegovo dimenzijo,

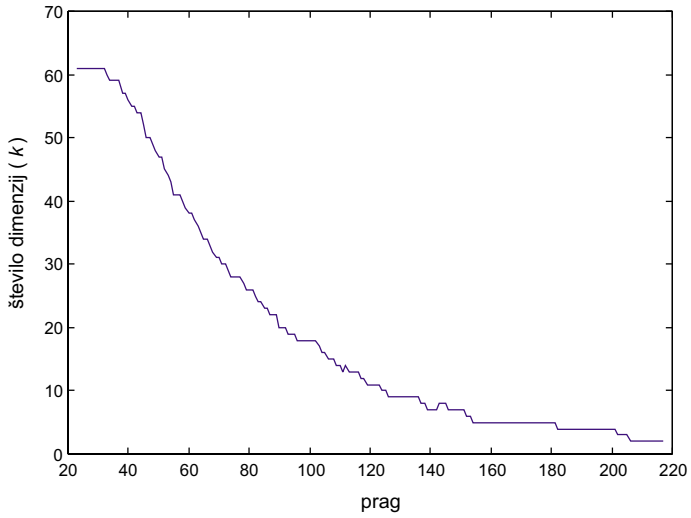
povzročimo minimalne perturbacije. Zato pričakujemo, da napake ne bodo velike. Primerjavo lahko naredimo tako, da izračunamo razdalje med projekcijami v podprostor in izvirnimi slikami. Vrednosti nato primerjamo pri podprostoru velikosti k, ki ga dobimo s paketno in postopno metodo.

Paketna metoda nam v eni sami iteraciji zgradi podprostor velikosti, ki je enaka številu slik n, nato pa ga skrajšamo na želeno velikost k. Postopna metoda pa potrebuje n iteracij, preden zgradi končni podprostor. Šele tega lahko primerjamo s tistim, ki ga izgradi paketna metoda. Velikost končnega podprostora je odvisna od kriterija za povečevanje dimenzij. Ta je lahko tak, da vnaprej določi končno število lastnih vektorjev, podprostor pa se povečuje samo do takrat, ko doseže to vrednost. Če pa je kriterij vezan na napako, ki jo povzročimo z ohranitvijo velikosti podprostora, pa ni mogoče vnaprej vedeti, kakšna bo končna velikost.

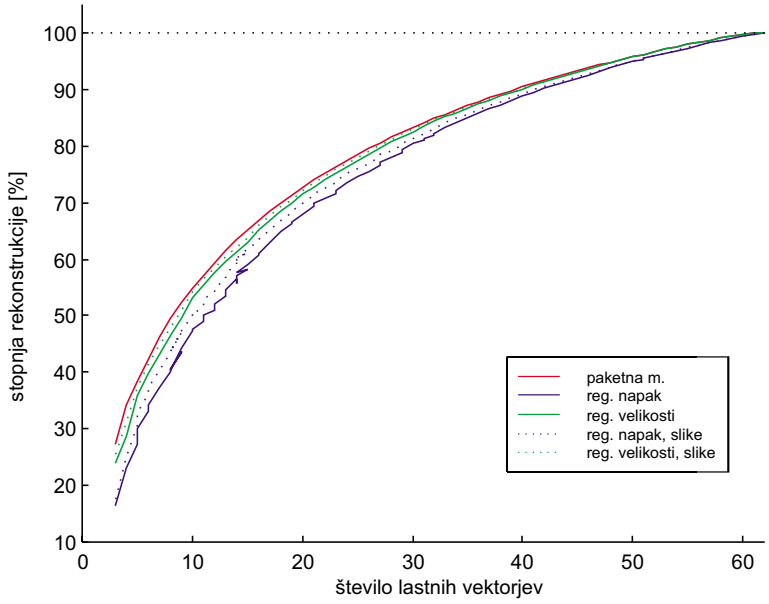
Rezultate želimo primerjati za vseh n možnih velikosti lastnega prostora. Za podprostore paketne metode lahko ves čas uporabljamo ustrezne lastne vektorje, ki jih metoda zgradi v prvi iteraciji. Postopne metode pa regulirajo velikost podprostorov med izgradnjo, kot smo opisali v razdelku 2.3.4. Zato je velikost podprostora po koncu izgradnje odvisna od izbora kriterija, ki ga postopek uporablja, in vrednosti pripadajočega praga. S postopno metodo moramo torej ponoviti izvajanje učenja na istem zaporedju n slik ob različnih pragih, dokler ne dobimo vsaj enega primereka za vsako velikost podprostora k, $k = 1 \ldots n$. Slika 4.3 prikazuje končne velikosti lastnih prostorov za $n = 62$, pri čemer je bilo za vsako točko na grafu potrebno eno izvajanje učenja iz n slik. Ker je število dimenzij diskretna vrednost, lahko dosežemo z različnimi pragi enako končno velikost. Na grafu opazimo tudi rahlo nemonotonost krivulje. Ti podprostori si torej niso povsem enakovredni, čeprav je število baznih vektorjev enako, vendar bomo razlike zanemarili in za vsak k obravnavali po en reprezentativen podprostor.

V naših eksperimentih smo torej za vsak k dobili po tri podprostore. Prvi je bil zgrajen s paketno metodo in nato skrajšan na dolžino k, drugega nam je zgradila postopna metoda, pri čemer smo se med iteracijami odločali za povečanje števila lastnih vektorjev na podlagi kriterija (2.36). Tretji podprostor pa je zgradila postopna metoda, ki je povečevala velikost podprostora, dokler ni dosegel velikosti k, nato pa je do konca ohranjal to velikost.

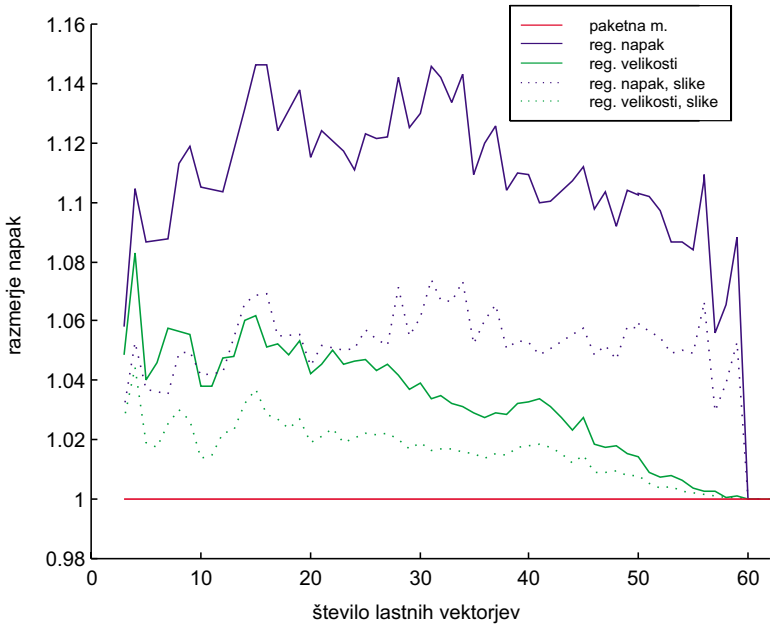
Vsakega od podprostorov smo zgradili iz istega zaporedja $n = 62$ slik $\mathbf{x}_i$. V podprostor, zgrajen s paketno metodo, smo projicirali vse slike naenkrat. Postopni metodi pa sta vsako od slik projicirali v ustrezni iteraciji, projekcije predhodno projiciranih slik pa preslikali v popravljeni podprostor. Kumulativno napako smo



Slika 4.3: Odvisnost velikosti podprostora od vrednosti praga.



Slika 4.4: Primerjava stopnje rekonstrukcije pri podprostorih prostorih različnih velikosti za paketno metodo, postopno metodo z omejevanjem napake („reg. napak“) ter postopno metodo z omejevanjem števila lastnih vektorjev („reg. velikosti“). Črtkani krivulji predstavljata rezultate, kjer slike projiciramo na koncu gradnje.



Slika 4.5: Razmerje velikosti rekonstrukcijske napake med posamezno postopno metodo ter paketno metodo.

potem za vsakega od podprostorov izračunali kot vsoto kvadratov rekonstrukcijskih napak

$$E_{\text{kum}} = \sum_{i=1}^{n} \|\mathbf{x}_i - (\mathbf{U}\mathbf{a}_i + \bar{\mathbf{x}})\|^2 \ . \tag{4.1}$$

Stopnjo rekonstrukcije nato izrazimo kot

$$R = \frac{E_{\text{max}} - E_{\text{kum}}}{E_{\text{max}}} \, , \tag{4.2}$$

pri čemer je E_{max} kumulativna največja rekonstrukcijska napaka, izražena kot

$$E_{\text{max}} = \sum_{i=1}^{n} \|\mathbf{x}_i - \bar{\mathbf{x}}\|^2 \ . \tag{4.3}$$

Slika 4.4 prikazuje vrednosti R v odvisnosti od velikosti podprostora za vsako od treh metod izgradnje. Oznake krivulj so izbrane glede na količino, ki jo regulirajo (regulacija velikosti lastnega prostora ali rekonstrukcijske napake). Paketna metoda predstavlja optimum in ima zato tudi največjo stopnjo rekonstrukcije. Sledi ji postopna metoda, ki povečuje podprostor do izbrane velikosti, nato pa to velikost vzdržuje do konca. Krivulja ima na grafu oznako „reg. velikosti“. Najslabši rezultat

dobimo s postopno metodo, pri kateri se za povečevanje odločamo glede na kumulativno napako. Njene rezultate označuje krivulja „reg. napak“. Grafu smo dodali še krivulji, ki prikazujeta rekonstrukcijske napake učnih slik, ki jih projiciramo v podprostor šele, ko je ta zgrajen do konca („reg. velikosti, slike“ in „reg. napak, slike“). Tako lahko vidimo vpliv perturbacij, ki jih izvajamo s projekcijami med vzdrževanjem velikosti podprostora. Rezultati kažejo, da je ta vpliv precej majhen, taka napaka pa je zelo nizka cena za občutno večjo smotrnost uporabe pomnilnika.

Postopne metode po pričakovanju ne delujejo bistveno slabše kot paketna metoda. Neposredno primerjavo prikazuje slika 4.5, kjer vsaka od krivulj predstavlja razmerje med kumulativno rekonstrukcijsko napako podane metode in napako paketne metode. V povprečju sta postopni metodi slabši za 3,25% oz. 10,62%.

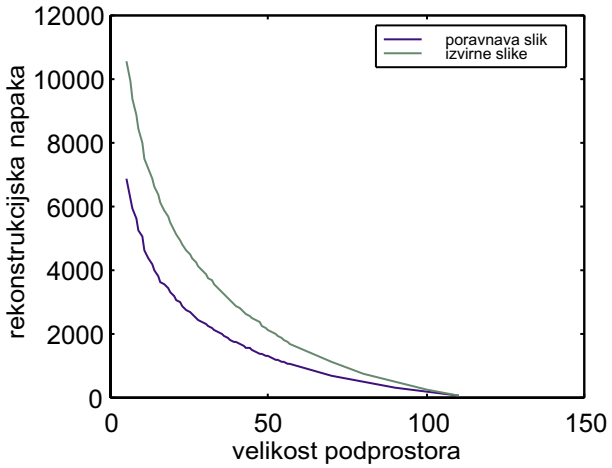
Razlika med kriterijema, ki smo jih uporabili pri postopnih metodah, se kaže v tem, da boljša izmed njih uporabi prvih k slik v celoti za svojo izgradnjo. Te potem prispevajo k takšnemu podprostoru, ki zelo dobro modelira preostale slike, nadaljnje perturbacije so zato manjše. Ta rezultat nam pove, da je bolje uporabljati to metodo, če želimo imeti na koncu lastni prostor velikosti izbrane k. V tem primeru je s stališča števila računskih operacij bolje iz prvih k slik izračunati podprostor s paketno metodo, nadaljevati pa z dopolnjevanjem podprostora tako, da ves čas ohranjamo velikost k.

Če pa nismo prepričani, kako velik končen podprostor želimo imeti, pa tudi metoda, ki se odloča glede na kumulativno napako, daje dovolj dobre rezultate. Eksperiment je pokazal, da dobimo primerljive rezultate s paketno metodo, če uporabimo lastni prostor, ki je večji za okoli pet dimenzij od tistega, ki ga zgradi paketna metoda. Takšna predstavitev slik s podprostorom pa je v primerjavi s potrebo paketne metode po hranjenju vseh slik še vedno bistveno bolj optimalna glede porabe pomnilnika.

4.2.1 Poravnava slik

V razdelku o učenju prostora iz panoramskih slik smo ugotovili, da lahko sicer tridimenzionalen problemski prostor lokacije in rotacije predstavimo tako, da na vsaki lokaciji shranimo le po enega predstavnika, ostale pa ustvarimo umetno z zamikanjem stolpcev slike. Pri učenju se moramo zato odločiti, katero rotacijo vzeti za reprezentančno.

Opisali smo tudi postopek izbora reprezentančne slike, ki poteka tako, da panoramsko sliko najprej zavrtimo tolikokrat, kolikor stolpcev ima slika. Za reprezen-

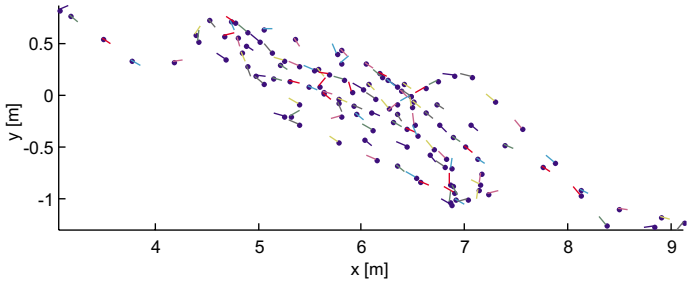


Slika 4.6: Primerjava rekonstrukcijskih napak postopne metode učenja, ki pred dodajanjem slike poišče najboljšo rotacijo (modra krivulja), in metodo, ki dodaja slike takšne, kot jih zajame robot (zelena krivulja).

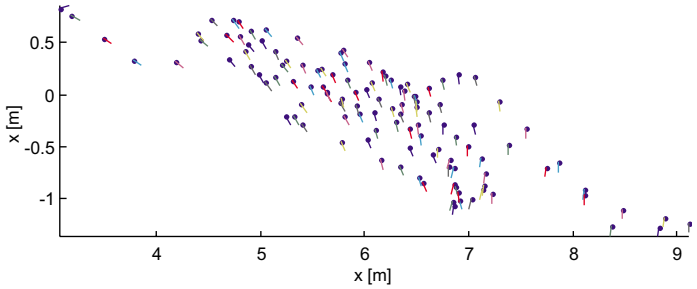
tančno sliko nato izberemo tisto rotacijo, ki je najbližja izgrajenemu podprostoru glede na velikost rekonstrukcijske napake. S tem zagotovimo, da se sistem uči iz tistih različic slik, ki se najbolje prilegajo trenutno izgrajenemu modelu. Takšna tehnika hkrati pripomore tudi k temu, da se med dopolnjevanjem lastni prostor čim manj spreminja.

Poravnavo slik smo preizkusili na zaporedju, ki smo ga zajeli z robotom, ki se je naključno premikal po prostoru in je bila orientacija kamere pri tem poravnana s smerjo vožnje. Slika 4.7 a) prikazuje diagram točk, v katerih so bile zajete posamezne slike, črtice pa nakazujejo usmeritev robota v teh točkah. Po učenju dobimo poravnane slike, katerih usmeritve prikazuje slika 4.7 b). Slika 4.7 c) pa prikazuje podmnožico slik po poravnavi in tik preden jih je uporabil za dopolnjevanje podprostora.

Vidimo, da algoritem slike poravnava na precej intuitiven način, saj na sliki 4.7 c) lahko zaznamo elemente prizora, ki med seboj v veliki meri sovpadajo. Zaradi takšne poravnave je stopnja podobnosti med zaporednimi slikami ohranjena v največji meri, hkrati pa tudi diagram lokacij in novih usmeritev kaže na konsistenco izbire poravnave. Preizkušeni algoritem omogoči tudi boljšo predstavitev shranjenih slik, kot če bi bile reprezentančne slike izbrane poljubno. To ugotovitev potrdi tudi slika 4.6, ki prikazuje rekonstrukcijske napake za metodo, ki slike pred dopolnjevanjem poravnava, in metodo, ki neposredno uporabi vhodno sliko. Rekonstrukcijska napaka na tej sliki je v odvisnosti od velikosti lastnega prostora, pri čemer smo upo-



a)



b)

c)

Slika 4.7: Poravnava slik pri učenju. Diagram lokacij, kjer so bile slike zajete ter izvirnih usmeritev (a), diagram usmeritev po poravnavah (b) in izgled slik po poravnavi.

rabili za učenje 118 slik (slika 4.7 a). Zaradi uporabljenega postopka so si namreč poravnane slike v veliki meri podobne, čeprav se lahko izvirne slike zaradi poljubnih rotacij med seboj občutno razlikujejo, četudi so posnete na isti lokaciji. Množico slik, ki so si med seboj precej podobne, pa podprostor majhnih dimenzij opiše bistveno bolje kot enako veliko množico bolj raznolikih slik.

4.3 Lokalizacija

Z naslednjim sklopom poskusov želimo ugotoviti, kako natančna je vizualna lokalizacija ob uporabi opisanih postopkov učenja in razpoznavanja.

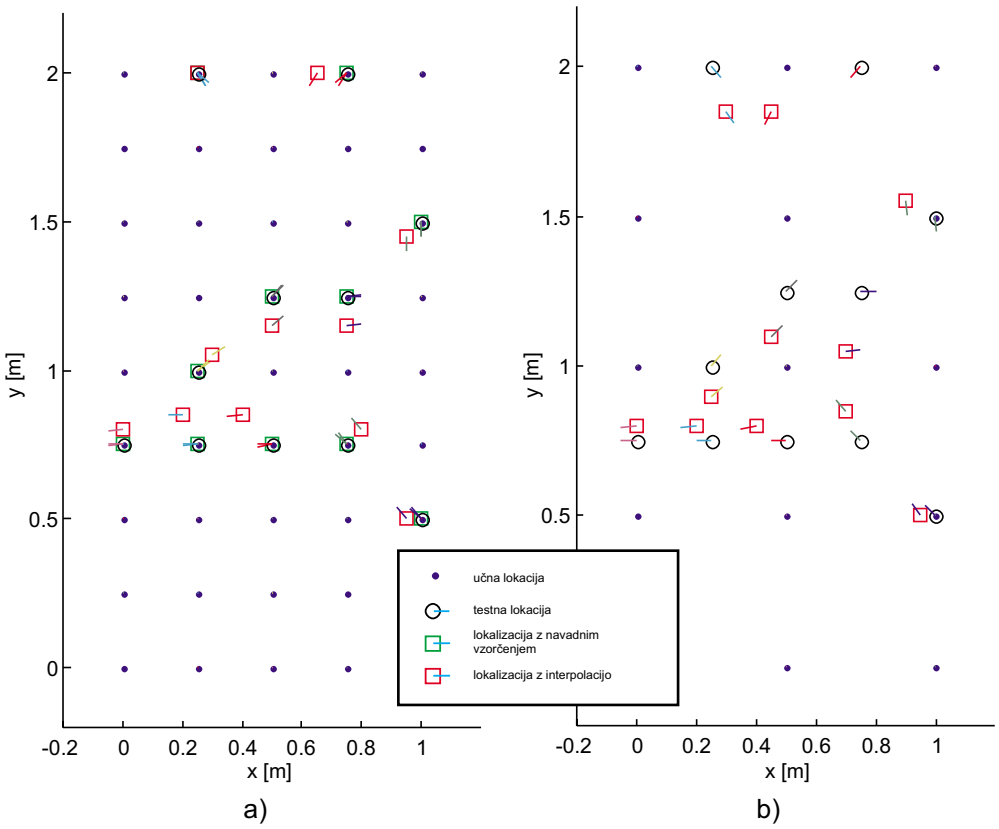
Za te poskuse smo pripravili dva ločena nabora slik, nabor učnih in testnih slik. Učne slike uporabimo za izgradnjo podprostora, testne slike pa so tiste, s katerimi izvajamo lokalizacijo. Ker merimo natančnost lokalizacije, smo pri obeh naborih natančno izmerili lokacije, kjer smo slike zajeli.

Lokacije učnih slik smo izbrali na ogliščih kvadratov velikosti 25 cm, ki tvorijo mrežo velikosti 5×9 lokacij. Robot je bil na vsaki lokaciji enako usmerjen. Tem 45 učnim slikam smo pridružili še 11 testnih slik, ki smo jih ponovno zajeli na izbranih lokacijah, spremenili smo jim le usmeritev. Slika 4.8 a) prikazuje postavitev tega poskusa. Na njej modre pike označujejo učne lokacije, črni krogi pa označujejo testne lokacije. Črtice iz središča krogov nakazujejo smer, kamor je bil robot na tej lokaciji obrnjen.

Za gradnjo podprostora smo izbrali metodo učenja, ki sproti poravnava slike, saj smo zanjo že v prejšnjem razdelku pokazali, da je superiorna. Poleg tega je učenje s to metodo hitro in zato primerno tudi za platformo mobilnega robota.

Najprej smo uporabili vse učne slike in na rezultatu uporabili lokalizacijo s testnimi slikami. Nato smo izvedli interpolacijo projekcij slik s korakom 5 cm glede na parametrični prostor. Dobili smo približek gostejšega vzorčenja prostora. Lokalizacijo smo ponovili s testnimi slikami.

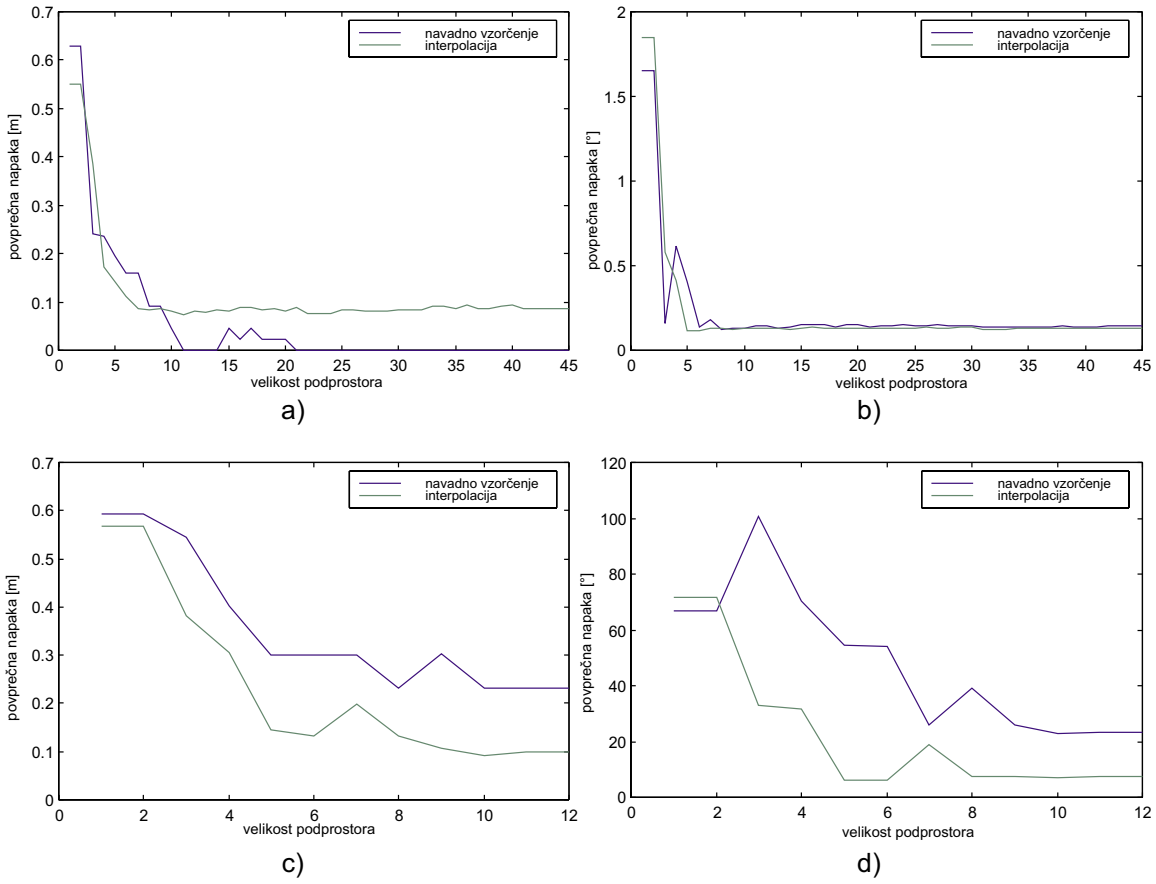
Glede na to, da lokacije testnih slik sovpadajo z lokacijami določenih učnih slik, je zanimiveje preizkusiti sposobnost generalizacije metod s podprostori. Mi smo to dosegli tako, da smo učno množico razredčili iz 25 cm na razdaljo 50 cm med parom sosednjih lokacij. Nabora testnih slik pri tem nismo spreminjali. S tem je le majhen del testnih slik sovpadel z lokacijami učnih slik. Postavitev tega dela poskusa prikazuje slika 4.8 b). Oznake na sliki so enake kot tiste na sliki a), vidimo le bistveno redkejšo razporeditev učnih lokacij. Za učenje smo uporabili isto metodo kot prej in ponovno izvedli interpolacijo s korakom 5 cm.



Slika 4.8: Postavitev učnih in testnih lokacij ter lokacije, ki jih dobimo s postopki lokalizacije pri polnem naboru učnih slik (a) in ob zmanjšanem naboru učnih slik (b).

Lokalizacija testne slike izbere najbolj podobno učno sliko oz. njen interpolirani približek in vrne pripadajoče koordinate. Napako v lokalizaciji izmerimo kot razdaljo med točno lokacijo testne slike in ocenjeno lokacijo. Napako v rotaciji označimo kot razliko med točno in ocenjeno rotacijo. Na sliki 4.9 so prikazane krivulje povprečne lokalizacijske napake v odvisnosti od velikosti uporabljenega podprostora.

Sliki 4.9 a) in b) prikazujeta rezultate lokalizacije, kjer smo za učenje uporabili vseh 45 slik. Krivulje dosežejo zadovoljivo vrednost že okoli velikosti 10 lastnih vektorjev in se potem ustalijo. Lokalizacija brez interpolacije je potem v veliki večini primerov točna, kar je v skladu s pričakovanji glede na sovpadanje testnih in učnih lokacij. Zanimivo je, da z interpolacijo te točnosti ne ohranimo, vendar je rezultat še vedno v okviru, ki je manjše od polovice izvirnega vzorčenja, kar razumemo kot dober rezultat. Slika 4.8 a) prikazuje diagram te lokalizacije za velikost podprostora 13. Z zelenimi kvadratki so označeni rezultati lokalizacije brez interpolacije, z rdečimi



Slika 4.9: Lokalizacijska napaka pri polnem naboru učnih slik (a) in pri zmanjšanem naboru učnih slik (b). Napaka je izražena kot povprečna razdalja med testnimi lokacijami in rezultati lokalizacije (a in c) ter razlika med koti (b in d). Vrednosti so v odvisnosti od velikosti podprostora.

kvadratki pa rezultati lokalizacije, ki uporablja interpolirane podatke. Črtica iz središča kvadratka označuje ugotovljeno smer, barve črtic pa sovpadajo s črticami ustreznimi testnimi smeri.

Če prostor vzorčimo preveč redko, je natančnost lokalizacije slik iz vmesnega, nevzorčenega prostora, omejena z gostoto vzorčenja. Vendar pa rezultati, ki jih prikazujeta sliki 4.9 c) in d), kažeta, da je v povprečju še vedno relativno zanesljiva. Podobno kot v prejšnjem primeru so rezultati, ki jih lahko že uporabimo, pri velikosti podprostora 10. Povprečna napaka je v tem območju manj kot 25 cm, kar je spet zelo dober rezultat glede na gostoto učenja po 50 cm. Vendar pa je v tem primeru uporaba interpolacije priporočljiva, saj lokalizacijo precej izboljša. Slika 4.8 b) predstavlja diagram lokalizacije v tem primeru. Zaradi večje jasnosti prikazuje le

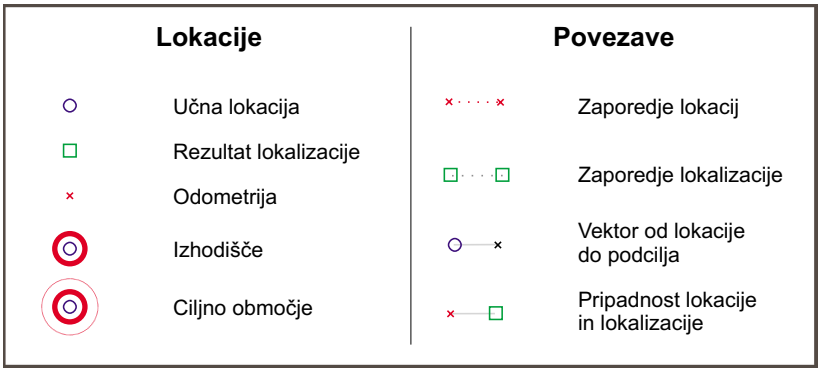
lokalizacijo z uporabo interpolacije.

Predstavljeni rezultati torej nakazujejo na veliko uporabnost vizualne lokalizacije, ki je, čeprav omejena na diskretne lokacije, še vedno dovolj natančna. Če potrebujemo višjo stopnjo natančnosti, pa lahko izvedemo navidezno vzorčenje z interpolacijo po parametričnem prostoru. Natančnost sicer potem ne doseže izbranega koraka interpolacije, vendar so približki vseeno zelo dobori.

4.4 Navigacija

V predhodnem razdelku smo se prepričali, da vizualna lokalizacija deluje dovolj natančno, da jo lahko uporabimo pri navigaciji. V nadaljevanju jo bomo zato uporabili v poskusih, katerih namen je ugotoviti, kako v praksi deluje osnovna navigacija do izbranega cilja.

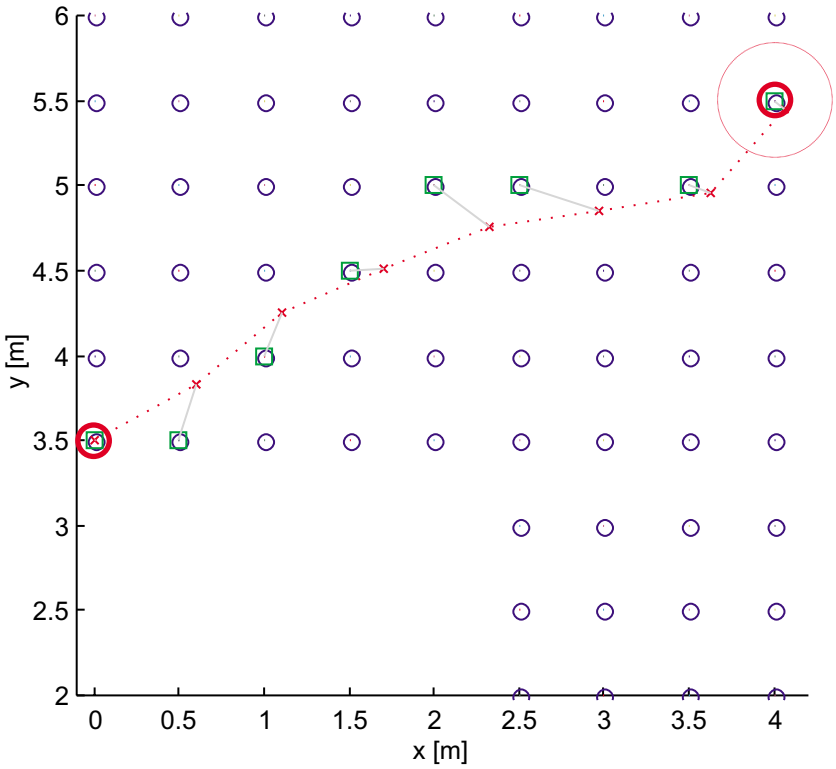
Za prvi del poskusov smo robota vodili na hodniku po mreži, katere celice so bile kvadrati velikosti 50 cm. Na robovih kvadratov smo zajeli po eno sliko, pri čemer so bile vse slike obrnjene v isto smer. Tako smo dobili 82 različnih slik, ob vsaki sliki pa smo shranili še razdaljo v smeri x in y od izbranega izhodišča.



Slika 4.10: Pomen oznak na slikah navigacije.

Te slike smo potem uporabili za učenje podprostora, ki smo ga zgradili postopno. Robota smo potem postavili na eno od učnih mest, ki ga je moral robot ugotoviti z lokalizacijo. S korakom največ 70 cm je robot potem navigiral med posameznimi podciliji do končnega cilja, ki je bilo iz izhodišča dosegljivo brez vmesnih ovir.

Nabore učnih slik drugega dela poskusov smo zajeli tako, da smo robota prosto vodili po prostoru, robot pa je periodično zajemal slike in jih shranjeval skupaj z vrednostjo vektorja odometrije. Lokacije so bile tako po prostoru porazdeljene neenakomerno in brez pravega vzorca, vključevale so neznano veliko napako ter



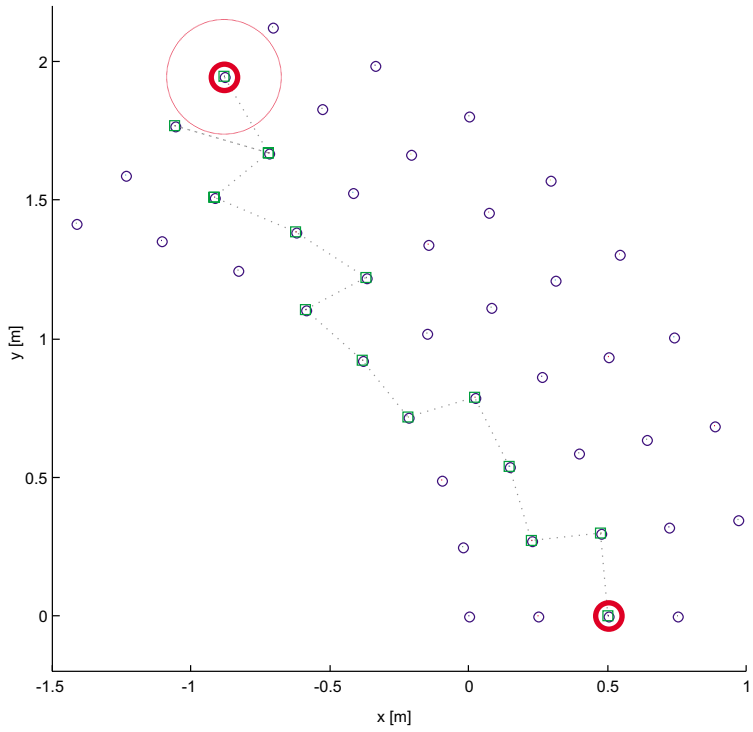
Slika 4.11: Navigacija po hodniku, učne slike razporejene v kvadratno mrežo.

različno usmerjene slike. Tudi te slike smo uporabili za učenje. Navigacija je nato potekala podobno kot pri prvem delu poskusov, le da je bil korak do novega podcilja manjši.

V nadaljevanju bomo rezultate opisali in prikazali z diagrami. Vsi diagrami imajo enotne oznake, ki predstavljajo različne vrste lokacij. V učnih lokacijah smo zajeli učne slike, odometrija robota pa prikazuje zaporedje lokacij, obiskanih med postopkom navigacije. Posebej sta označeni izhodiščna in končna učna lokacija. Vsaki lokaciji odometrije pripada ena učna lokacija, ki jo je postopek vrnil z lokalizacijo pripadajoče slike. Oznake prikazuje slika 4.10.

Slika 4.11 prikazuje postavitev in rezultat poskusov na hodniku ob enakomerno porazdeljenih učnih lokacijah. Vidimo, da je bila začetna lokalizacija točna, ostale na poti do cilja pa so bile dovolj blizu, da je robot pripeljal na cilj v najmanjšem številu korakov.

Ko smo učne lokacije postavljali enakomerno po prostoru, smo za vsako lokacijo poskrbeli, da je bila res v lokaciji, katere koordinate smo pripisali pripadajoči sliki. Med prosto vožnjo pa se zanašamo samo na podatke, ki nam jih vrača odometrija.

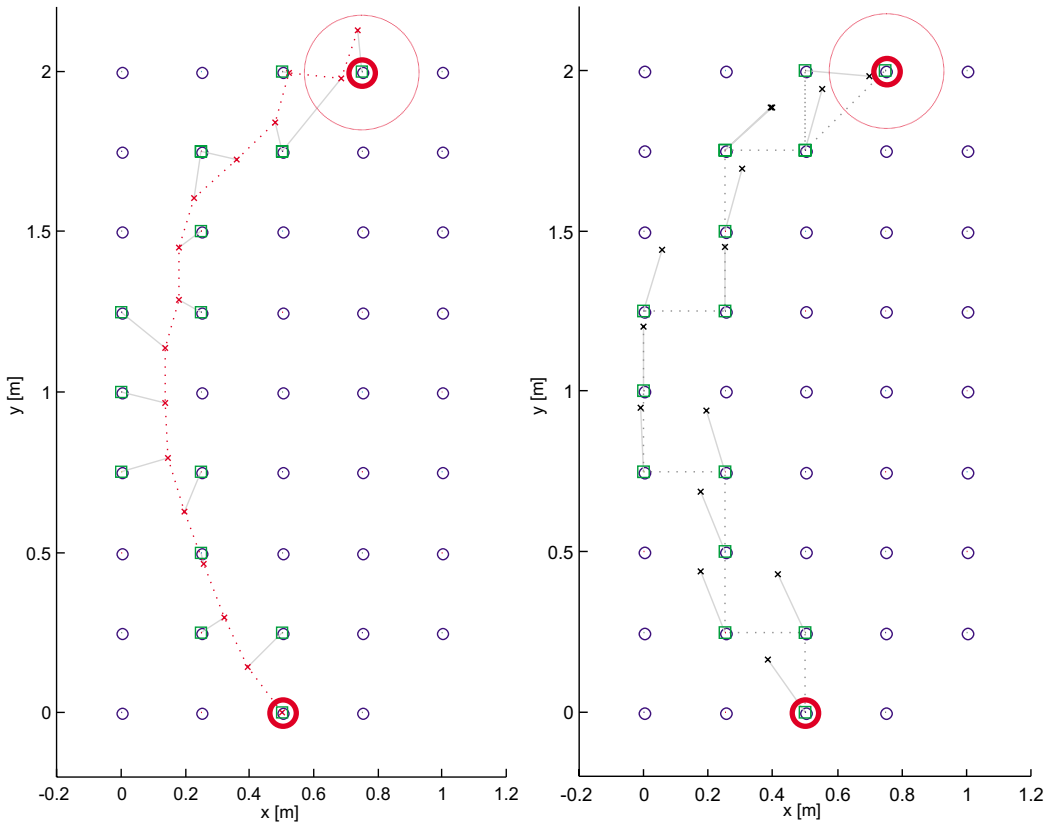


Slika 4.12: Razpored učnih lokacij po umetnem ukrivljanju, izhodiščna in ciljna lokacija ter zaporedje lokalizacij med navigacijo.

Ta je na krajše razdalje relativno točna, sčasoma pa se napaka akumulira. Zanimalo nas je, kako uporaben za navigacijo je zemljevid, ki vsebuje takšno napako. To smo preverili tako, da smo za učno množico uporabili nabor slik, ki smo ga v prejšnjem razdelku uporabili za poskuse z lokalizacijo. Oznake lokacij smo nato umetno spremenili tako, da smo dobili mrežo ukrivljeno za 45°. Rezultat ukrivljanja prikazuje slika 4.12.

Robota smo nato postavili v eno od lokacij ter za cilj navigacije izbrali takšno lokacijo, ki je tudi v ukrivljenem prostoru dosegljiva, ne da bi pot zašla izven naučenega območja, kjer so ovire. Korak do naslednjega podcilja je bil dolg največ 30 cm. Posledica umetnega ukrivljanja je, da je na vsakem koraku naslednji podcilj v napačni smeri. Slika 4.13 prikazuje takšno izbiro podciljev relativno glede na lokacije, ki jih določi z lokalizacijo.

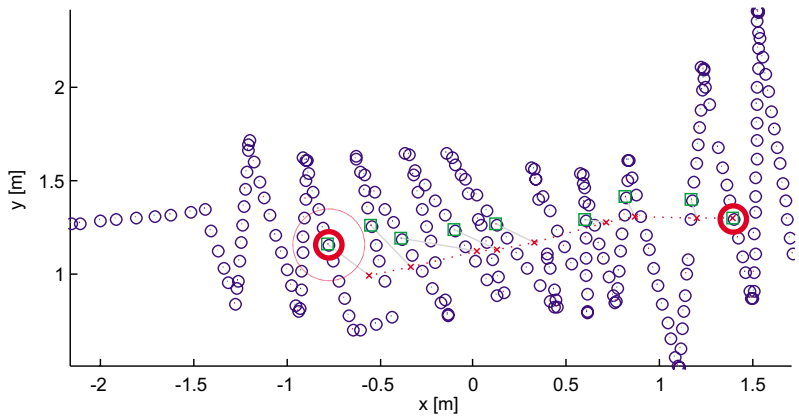
Celotni potek navigacije skupaj z označenim zaporedjem robotove odometrije, pripadajočimi lokalizacijami ter smermi naslednjih korakov prikazuje slika 4.13. Na sliki so prikazane prave učne lokacije, slika 4.12 pa služi kot primerjava, saj vsebuje prostor, kot ga ima predstavljenega robot. Vidimo lahko, da je sprotno preverjanje z



Slika 4.13: Potek navigacije skozi prostor s slike 4.12, prikazan z resničnimi učnimi lokacijami. Leva slika prikazuje zaporedje lokacij, razbranih iz odometrije, ter pripadajoče lokalizacije, desna slika pa prikazuje usmeritev do podciljev iz vsake lokalizirane lokacije.

lokalizacijo v podciljih ključno za uspešno navigacijo. Čeprav je odometrija tako med učenjem kot med navigacijo natančna le v omejenem obsegu, pa nam lokalizacija poda potrebno povratno informacijo, ki vodi in popravlja navigacijo. Z rezultati tega poskusa smo ugotovili, da je navigacija izvedljiva tudi s pomočjo slik, ki smo jih zajeli s prosto vožnjo in si lokacije beležimo le iz odometrije.

Sliki 4.14 in 4.15 prikazujeta potek navigacije po prostoru, ki smo se ga naučili med prosto vožnjo. Navigacija se uspešno zaključi, ko lokalizacija vrne lokacijo, ki je v bližnji okolici ciljne lokacije. Prostor s slike 4.14 je bil isti del laboratorija, kot smo ga za navigacijo uporabljali pri poskusu s slike 4.12. Vzorčenje je bilo zato gostejše. Iz poteka navigacije vidimo, da je lokalizacija v skladu s smerjo vožnje in globino, do katere je robot v danem trenutku napredoval. Vidno odstopanje med odometrijo in pripadajočimi lokalizacijami je posledica že omenjenih napak med raziskovanjem



Slika 4.14: Navigacija v laboratoriju, učne slike zajete med prosto vožnjo.

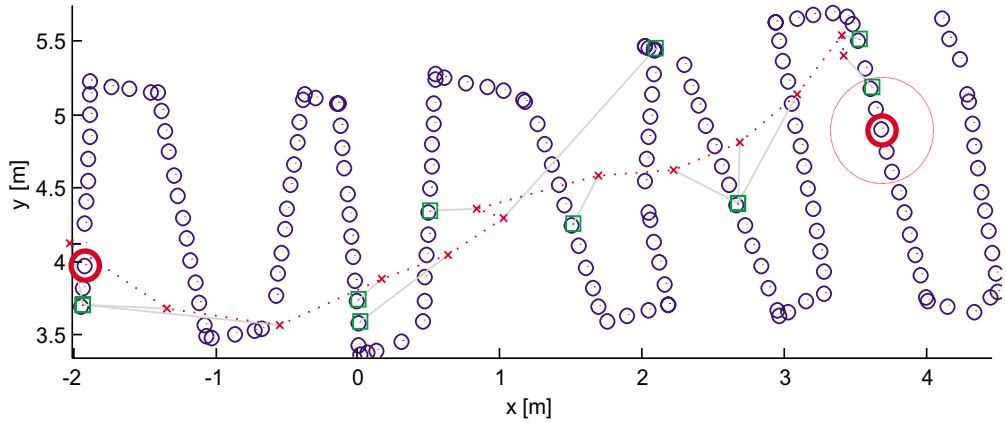
in navigacijo in ni moteče.

Pri raziskovanju hodnika smo uporabili večje razdalje, kot jih prikazuje slika 4.15. Takšen prostor pa zahteva nekoliko previdnejšo izbiro podciljev, saj lahko pristane v vmesnem prostoru, ki ni pokrit z učnimi lokacijami. Zato smo postopek izbire podcilja nekoliko razširili. Potem ko smo na zgoraj opisani način določili točko, smo poiskali njej najbližjo učno lokacijo izmed primernih kandidatov. Kandidati so pri tem vse učne lokacije, ki iz izhodiščne lokacije ne predstavljajo odklona od poti, ki je večji od izbranega kota. Tako smo si zagotovili, da je podcilj v pokritem območju in da hkrati robot ne skrene iz prvotne poti v preveliki meri.

Navigacija se je tudi tokrat uspešno zaključila v bližini ciljne lokacije. Vidimo sicer nekaj več napak lokalizacije, ki povzročijo, da se robot usmeri v napačno smer. Vendar pa v večini primerov naslednja lokalizacija povrne navigacijo v pravo smer.

Takšne napake so bile posledica uporabe globalne lokalizacije (tj. iskanje najbližjega soseda izmed vseh učnih slik) tudi potem, ko se robot pripelje do podciljev. Pričakujemo lahko, da bi se takšne napake drastično zmanjšale, če bi takrat omejili kandidate le na ožjo okolico zadnjega podcilja, premaknjeno za vektor navigacije do trenutnega podcilja.

Zanimivo je to, da je natančnost lokalizacije visoka kljub relativno redkemu vzorčenju prostora. Malo verjetno je, da bi ob koncu navigacije do podcilja robot zavzel točno enako lokacijo, kot jo je med učenjem. Če bi kljub temu želeli povečati natančnost lokalizacije, lahko to dosežemo z interpolacijo med točkami v podprostoru.



Slika 4.15: Navigacija po hodniku, učne slike zajete med prosto vožnjo po prostoru, velik razkorak med slikami v smeri navigacije.

Poglavje 5

Zaključek

Predstavili smo sistem, ki omogoča lokalizacijo in navigacijo avtonomnega mobilnega robota s pomočjo vizualne informacije. Razvili smo metode, ki iz množice slik zgradijo model prostora, ki ga robot razišče, ta model pa uporablja za vizualno navigacijo.

Glavna značilnost razvitih postopkov je uporaba izgleda kot osnovne enote vizualne informacije. To pomeni, da slik ne obdelujemo tako, da bi iz njih izbrali samo podmnožico slikovnih elementov ali ugotavljali strukturo prizora. Prednost takega postopka je, da uporabimo slikovno informacijo v takšni obliki, kot jo dobimo iz vizualnega senzorja. Poleg tega ne podajamo predpostavk o strukturi prizorov in prostorov, ampak smo sposobni izkoristiti katerokoli okolje, ki ima dovolj razgiban izgled.

Postopek, ki za svoje delovanje potrebuje izgled in primerjanje izgledov, zahteva predstavitev celih slik ali delno predstavitev, ki vsebuje najbolj bistvene dele informacije izvirnih slik. Slike pa zasedajo razmeroma velik del pomnilnika, ki bi ga sicer lahko uporabili za bolj dinamične podatke. Poleg tega na mobilnem robotu pomnilnika običajno ni veliko in je problematika smotrne porabe pomnilnika še bolj pereča. Zato smo razvili metodo, ki izkorišča postopno gradnjo podprostorov z uporabo metode glavnih komponent [9], pomnilnik pa uporablja le za razmeroma dinamične podatke spreminjajočega se modela prostora in predstavitve slik. Pokazali smo, da je uporaba naše metode upravičena, saj slabosti praktično nima.

Slike prostora se med seboj razlikujejo, če jih posnamemo z različnih lokacij. Zato lahko zberemo dovolj veliko končno število slik z različnih lokacij, jih med seboj povežemo po dostopnosti in jim zabeležimo lokacije, in dobili bomo zemljevid prostora. Kot smo pokazali, je takšen zemljevid mogoče uporabljati za navigacijo

po prostoru, četudi so lokacije na zemljevidu netočne. To se sklada s človeško predstavo o prostorih, saj imamo ljudje nek okvirni občutek o razsežnostih prostorov in razdaljah med lokacijami, ki pa ni nujno točen. Kljub temu smo sposobni uspešno navigirati med prostori, saj si sproti pomagamo z vizualno informacijo. Namesto da bi se v danem trenutku mehanično napotili na izbrani cilj, raje navigiramo progresivno, tako da se usmerimo v najbolj obetajočo smer ter to izvajamo, dokler ne pridemo dovolj blizu izbranega cilja. S poskusi smo pokazali, da se takšen pristop obnese tudi za mobilnega robota, katerega predstava o lokacijah je razmeroma netočna.

Predstavljeni postopki omogočajo izkoriščanje principa sočasne lokalizacije in izgradnje zemljevida (ang. *Simulataneous Localisation and Map Building*, SLAM) [18]. Glede na to, da lahko zemljevid dopolnjujemo z novimi slikami in lokacijami, namreč nismo vezani na fazo učenja, ki bi bila strogo ločena od faze uporabljanja znanja. Zato lahko v prostoru, ki smo ga delno že raziskali, izvedemo navigacijo do območja, ki bi ga radi še pokrili z zemljevidom, ter nove ugotovitve preprosto dodajamo zemljevidu. Nadalje lahko, če ugotovimo, da se je robot med navigacijo izgubil, prostor začnemo raziskovati znova. Rezultat bi bil nov podprostor, ki je od prejšnjega ločen. Med navigacijo pa lahko potem preverimo skladnost lokacij s tistimi iz prvotnega podprostora. Če se izkaže, da se zemljevida med seboj pokrijeta, ju lahko združimo v en zemljevid s postopkom združevanja podprostorov [10].

Slabost razpoznave iz celih pogledov je ta, da je postopek razmeroma občutljiv na spremembe v prostoru. Če se na sliki pojavijo elementi, ki zastirajo del prizora, ali se je od učenja spremenila osvetlitev, lahko pride do napak v razpoznavi. Te težave rešijo robustne metode za učenje in razpoznavo. Napake, ki nastanejo zaradi zastrtih delov prizora, odpravlja robustna metoda projekcije slike v podprostor, ki namesto vseh slikovnih elementov uporabi le podmnožico najbolj obetajočih kandidatov [15]. Težavam zaradi sprememb v osvetlivi se izognemo z uporabo filtrov, ki vrnejo sliko, ki je nedovisna od osvetlitve [14]. Če pa že med učenjem pričakujemo dinamične elemente, ki bi lahko zastirali dele prizora, pa uporabimo robustno učenje [30].

Metode, ki smo jih opisali, zelo dobro delujejo v predstavljenem okviru. Izzivi, ki ostajajo za prihodnje raziskave, pa vključujejo možnosti in metode za popravljanje in izboljševanje obstoječega zemljevida. Sem spada ugotavljanje stopnje neskladnosti delov zemljevida zaradi morebitnih večjih napak v lokacijah, usklajevanje ter izvajanje korekcij. Za slednje bi se verjetno obnesla metoda vzmeti, kot so jo uporabili v [8]. Odprto ostaja tudi vprašanje vizualnega raziskovanja neznanega prostora,

ki bi se s pomočjo opisanih metod samostojno odločal o tem, katere dele prostora mora robot še obiskati in kako preveriti in dopolniti že obiskana območja. Pri tem procesu bi robotu pomagal senzor za ocenjevanje razdalje do predmetov oz. bi iz panoramskih slik ugotavljal razdalje do ovir.

Zahvala

Zahvaljujem se prof. dr. Alešu Leonardisu za pomoč, ideje in vodstvo pri raziskavah ter pri nastanku magistrskega dela. Zahvalil bi se tudi dr. Danijelu Skočaju za pomoč pri nastanku tega dela, Matjažu Joganu in ostalim članom Laboratorija za računalniški vid za dobro in raziskovalno vzdušje, kolegom in prijateljem za moralno podporo ter staršem, ki mi stojijo ob strani.

Literatura

[1] Claus S. Andersen, Stephen D. Jones, and James L. Crowley. Appearance based process for visual navigation. In *5th Internation Symposium on Intelligent Robotic Systems*, pages 227–236. Royal Institute of Technology, Stockholm, Sweden, July 1997.

[2] A.A. Argyros, C. Bekris, and S. Orphanoudakis. Robot homing based on corner tracking in a sequence of panoramic images. In *Computer Vision and Pattern Recognition Conference (CVPR '2001), Hawaii, USA*, volume 2, pages 3–10, December 2001.

[3] Matej Artač, Matjaž Jogan, and Aleš Leonardis. Mobile robot localization using an incremental eigenspace model. In *IEEE International Conference on Robotics and Automation, May 11–15, 2002, Washington, D. C.*, volume 1, pages 1025–1030. IEEE, 2002.

[4] Paul Blaer and Peter Allen. Topological mobile robot localization using fast vision techniques. In *IEEE International Conference on Robotics and Automation, May 11–15, 2002, Washington, D. C.*, pages 1031–1037. IEEE, 2002.

[5] B. A. Cartwright and T. S. Collett. Landmark learning in bees. *Journal of Comparative Physiology A*, 151:521–543, 1983.

[6] Guilhwemw N. DeSouza and Avinash C. Kak. Vision for mobile robot navigation: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(2):237–267, February 2002.

[7] M.O. Franz, B. Schölkopf, H.A. Mallot, and H.H. Bülthoff. Where did I take that snapshot? Scene-based homing by image matching. *Biol. Cybern.*, 79(3):191–202, 1998.

[8] Verena Vanessa Hafner. Cognitive maps for navigation in open environments. In *Proceedings of the 6th International Conference on Intelligent Autonomous Systems*, pages 801–808. IOS Press, 2000.

[9] P. Hall, D. Marshall, and R. Martin. Incremental eigenalysis for classification. In *British Machine Vision Conference*, volume 1, pages 286–295, September 1998.

[10] P. Hall, D. Marshall, and R. Martin. Merging and splitting eigenspace models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(9):1042–1048, 2000.

[11] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:417–441, 1933.

[12] H. Ishiguro and S. Tsuji. Image-based memory of environment. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pages 634–639, 1996.

[13] Matjaž Jogan, Emil Žagar, and Aleš Leonardis. Karhunen-Loéve decomposition of a set of rotated templates. *IEEE Transactions on Image Processing, to appear*, 2003.

[14] Matjaž Jogan and Aleš Leonardis. Robust localization using an omnidirectional appearance-based subspace model of environment. *Robotics and Autonomous Systems, to appear*, 2003.

[15] Matjaž Jogan and Aleš Leonardis. Robust localization using panoramic view-based recognition. In *15th International Conference on Pattern Recognition*, volume 4, pages 136–139. IEEE Computer Society, September 2000.

[16] B. J. A. Kröse, N. Vlassis, R. Bunschoten, and Y. Motomura. A probabilistic model for appearance-based robot localization. *Image and Vision Computing*, 19(6):389–391, April 2001.

[17] Benjamin Kuipers. The spatial semantic hierarchy. *Artificial Intelligence*, 119(1–2):191–233, May 2000.

[18] J. J. Leonard and H. F Durrant-Whyte. Simultaneous map building and localization for an autonomous mobile robot. In *IEEE/RSJ International Workshop on Intelligent Robots and Systems*, pages 1442–1447, Osaka, Japan, November 1991.

[19] A. Leonardis, H. Bischof, and J. Maver. Multiple eigenspaces. *Pattern Recognition*, 35(11):2613–2627, 2002.

[20] H.A. Mallot, M.O. Franz, B. Schölkopf, , and H.H. Bülthoff. The view-graph approach to visual navigation and spatial memory. In *Proceedings of the 7th International Conference on Artificial Neural Networks ICANN97, Lausanne, Switzerland*. Springer-Verlag, 1997.

[21] Martin C. Martin and Hans Moravec. Robot evidence grids. Technical Report CMU-RI-TR-96-06, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, March 1996.

[22] Y. Matsumoto, M. Inaba, and H. Inoue. View-based approach to robot navigation. *Journal of Robotics Society of Japan*, 20(5), 2002.

[23] Yoshio Matsumoto, Masayuki Inaba, and Hirochika Inoue. Memory-based navigation using omni-view sequence. In Alexander Zelinskey, editor, *Field and Service Robotics*, pages 171–178. Springer, 1998.

[24] Emanuele Menegatti, Enrico Pagello, and Mark Wright. Using omnidirectional vision within the spatial semantic hierarchy. In *IEEE International Conference on Robotics and Automation, May 11–15, 2002, Washington, D. C.*, pages 908–914. IEEE, 2002.

[25] S. K. Nayar, S. A. Nene, and H. Murase. Subspace methods for robot vision. *IEEE Trans. on Robotics and Automation*, 12(5):750–758, October 1996.

[26] Tomáš Pajdla and Václav Hlaváč. Zero phase representation of panoramic images for image based localization. In Franc Solina and Aleš Leonardis, editors, *8-th International Conference on Computer Analysis of Images and Patterns*, number 1689 in Lecture Notes in Computer Science, pages 550–557. Springer Verlag, September 1999.

[27] Lucas Paletta, Simone Frintrop, and Joachim Hertzberg. Robust localization using context in omnidirectional imaging. In *IEEE International Conference on Robotics and Automation, May 21–26, Seul, Korea*, pages 2072–2077. IEEE, 2001.

[28] Luca Regini, Guido Tascini, and Primo Zingaretti. Appearance-based robot navigation. Technical report, Istituto di Informatica, University of Ancona, 2002.

[29] H. Sirovich and M. Kirby. Low-dimensional procedure for the characterization of human faces. *J. Opt. Soc. America A*, 4(3):519–524, 1987.

[30] Danijel Skočaj. *Robust Subspace Approaches to Visual Learning and Recognition.* PhD thesis, University of Ljubljana, Faculty of computer and information science, 2003.

[31] Simon Thompson and Alexander Zelinsky. Accurate local position using visual landmarks from a panoramic sensor. In *IEEE International Conference on Robotics and Automation, May 11–15, 2002, Washington, D. C.*, pages 1025–1030. IEEE, 2002.

[32] Emanuele Trucco and Alessandro Verri. *Introductory Techniques for 3-D Computer Vision.* Prentice Hall, 1998.

[33] N. Winters, J. Gaspar, G. Lacey, and J. Santos-Victor. Omni-directional vision for robot navigation. In *Proc. IEEE Workshop on Omnidirectional Vision - Omnivis00*, pages 21–27, 2000.

Izjava

Izjavljam, da sem magistrsko delo izdelal samostojno pod vodstvom mentorja prof. dr. Aleša Leonardisa. Izkazano pomoč drugih sodelavcev sem v celoti navedel v zahvali.